МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА

В. А. Толстохатько
О. Є. Поморцева
І. М. Патракеєв

БАЗИ ДАНИХ: ПРОЕКТУВАННЯ ТА ВИКОРИСТАННЯ ДЛЯ ОБЛІКУ НЕРУХОМОГО МАЙНА

Рекомендовано Міністерством освіти і науки України як навчальний посібник для студентів вищих навчальних закладів, які навчаються за напрямом підготовки «Геодезія, картографія та землеустрій»

Харків
ХНУМГ
2014

УДК [330.47:004.65] (075)
ББК 65.22Ся73-6+32.973.26-018.2я73-6
Т52

Автори:

Толстохатько Віктор Антонович, проф.;
Поморцева Олена Євгенівна, к. т. н., доц.;
Патракеєв Ігор Михайлович, к. т. н., доц.

Рецензенти:

Янцевич А. А., докт. фіз.-мат. наук, професор кафедри математичних методів в економіці ХНУ ім. В. Н. Каразіна;
Колгатін О. Г., доктор пед. наук, професор кафедри інформатики ХНПУ ім. Г. С. Сковороди

Рекомендовано Міністерством освіти і науки України як навчальний посібник для студентів вищих навчальних закладів, які навчаються за напрямом підготовки «Геодезія, картографія та землеустрій» (лист № 1/11–7213 від 14.05.2014)

Толстохатько В. А.

Т52 Бази даних: проектування та використання для обліку нерухомого майна : навч. посібник / В. А. Толстохатько, О. Є. Поморцева, І. М. Патракеєв; Харк. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Х. : ХНУМГ, 2014. – 174 с.

ISBN 978-966-695-340-0

У посібнику розглянуто: загальні відомості про бази даних і системи керування базами даних, основні теоретичні поняття та терміни, які розкривають поняття баз даних, основи та принципи проектування реляційних баз даних, прийоми та засоби створення таблиць, запитів, форм і звітів бази даних в середовищі MS Access 2010, публікацію баз даних у мережі Internet, а також методику розробки додатків з використанням макросів і мови програмування Visual Basic for Application MS Access 2010.

УДК [330.47:004.65] (075)
ББК 65.22Ся73-6+32.973.26-018.2я73-6

ISBN 978-966-695-340-0

ЗМІСТ

Найкращому другу та вчителю присвячується

ВСТУП

Навчальний посібник підготовлений згідно з програмою нормативної навчальної дисципліни **«Бази даних»**, яка має професійне спрямування підготовки бакалаврів за напрямом «Геодезія, картографія та землеустрій».

Основні дисципліни, що забезпечують вивчення дисципліни «Бази даних», і дисципліни, що забезпечуються вивченням цієї дисципліни, подані на рис. 1.

У посібник включено навчальний матеріал, який необхідний студентам для якісного вивчення наступних наукоємних дисциплін: «Проектування баз геоданих», «Програмування геоінформаційних задач», «Основи геоінформаційних систем (ГІС)», «Технології ГІС», «Аналіз ГІС».

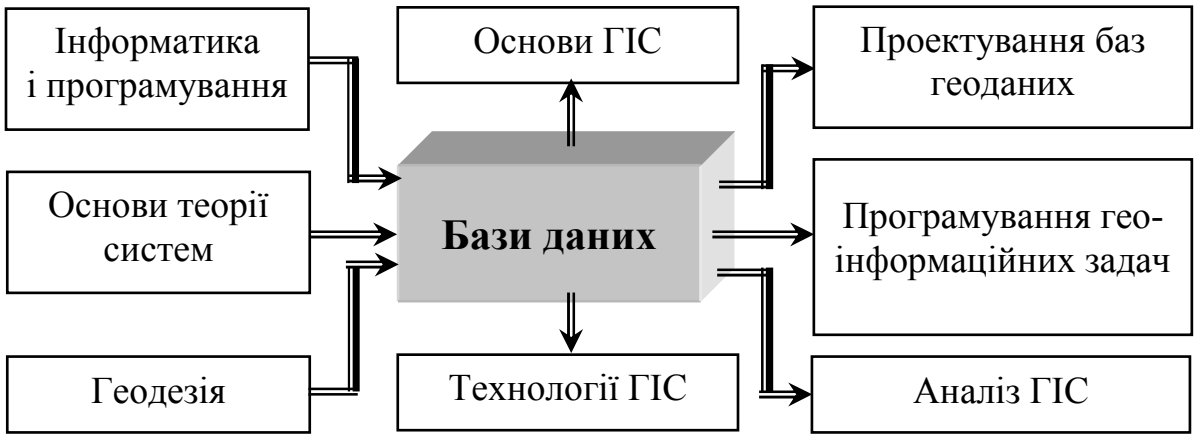


Рис. 1 – Місце дисципліни «Бази даних» у навчальному процесі

Навчальний посібник містить 4 розділи:

1. Загальні відомості про бази даних.
2. Проектування реляційних баз даних.
3. Створення реляційних баз даних із використанням СКБД Microsoft Access.
4. Завдання на лабораторні заняття та індивідуальне навчально-дослідне завдання.

У перший розділ включено загальні відомості про бази даних і системи керування базами даних (СКБД), основні теоретичні поняття й терміни баз даних, розглянуто склад реляційної та інфологічної моделей даних. Теоретичний матеріал та основні поняття баз даних подано з використан-

ням простих і наочних прикладів, які містять інформацію про протікання навчального процесу у вищому навчальному закладі. Ці приклади добре розуміє кожен студент.

У другому розділі розглянуто принципи, методи та етапи проектування реляційних баз даних. Наведено приклад проектування бази даних для вирішення прикладної задачі обліку нерухомого майна, яка відповідає предметній області студентів даного напряму підготовки. На цьому прикладі будується і матеріал третього розділу.

У третьому розділі розглянуто прийоми та засоби створення таблиць, запитів, форм і звітів бази даних у середовищі MS Access, публікацію баз даних у мережі Internet, а також методику розробки додатків із використанням макросів і мови програмування Visual Basic for Application (VBA) MS Access. Розробка проекту бази даних здійснюється із застосуванням задачі про облік нерухомого майна. За допомогою цієї задачі студенти відпрацьовують у повному обсязі методику розробки бази даних «Кадастр» – від постановки задачі до розробки додатка.

Четвертий розділ присвячено індивідуальному навчально-дослідному завданню студента. Подано варіанти завдань, основні вимоги до створюваних баз даних та наведено приклад виконання індивідуального завдання.

Професійні компетентності, що формуються під час вивчення навчальної дисципліни «Бази даних»

У процесі навчання студенти отримують необхідні знання під час лекційних занять, закріплюють і поглиблюють знання, набувають практичних навичок та вмінь при виконанні лабораторних робіт. Особливе значення має самостійна робота студентів, під час виконання якої вони самостійно розробляють індивідуальний проект прикладної бази даних. У процесі роботи над проектом студенти набувають навичок роботи з науково-технічною літературою, навчаються самостійно приймати рішення та робити висновки. У результаті засвоєння матеріалу навчального посібника у студента повинні сформуватися такі компетентності.

Проектні, що пов’язані з використанням основних принципів побудови реляційних баз даних, розробки та створення форм для вводу даних і їх аналізу, розробки та створення звітів.

Аналітичні, що пов’язані з використанням прикладних пакетів для аналізу предметної області у зазначені терміни засобами персональних

комп′ютерів, застосуванням одержаних відомостей для аналізу, самостійного вибору й освоєння нових програмних продуктів.

Управлінські, що пов’язані із застосуванням засобів сполучення різних прикладних пакетів для комплексної обробки геоінформаційних даних, у тому числі в разі віддаленого доступу до бази даних, та створенням супровідної документації.

Принципи, які покладено в основу побудови навчального посібника

В основу побудови посібника покладено принципи системності і практичної спрямованості з використанням сучасної СКБД MS Access.

Згідно з принципом системності питання розробки реляційних баз даних розглядаються в такому порядку: формулюється проблема, задачі, методи та засоби їх вирішення. Матеріал викладається за принципом «від простого – до складного». Такий підхід сприяє формуванню у студентів системних знань і розвитку у них творчого мислення.

Принцип практичної спрямованості передбачає фундаментальну наукову підготовку й активне практичне навчання студентів. Посібник містить як основні наукові уявлення, так і створює умови для формування навичок і вмінь за допомогою практичної форми навчання. У посібник включено теоретичні основи розробки інфологічних і реляційних моделей даних, методи проектування баз даних та практичні підходи до створення реляційних баз даних із використанням засобів СКБД MS Access. У роботі розглядається російськомовна версія MS Access, тому назви елементів управління, меню, команд і відповідних їм діалогових вікон наводяться російською мовою.

У посібнику зміст практичного навчання реалізовано на базі методу активної рефлексії. Застосування цього методу припускає вивчення й обмірковане повторення студентами операцій, які необхідні для досягнення поставленої мети. При викладанні матеріалу дається короткий опис теоретичних основ, аналіз способів застосування отриманого знання та детально розглядається процес розробки проекту бази даних засобами MS Access. Завдяки цьому студенти навчаються створювати бази даних самостійно з використанням матеріалу посібника.

Важливим є те, що в практичній частині посібника використовується єдина задача, яка має практичну спрямованість за напрямом підготовки. Студенти мають можливість прослідкувати процес створення проекту від

постановки задачі до її вирішення. Формування навичок може здійснюватися як під керівництвом викладача в аудиторії, так і вдома шляхом самостійного створення індивідуальних проектів на основі розглянутої методики розробки проекту.

При відборі матеріалу враховувалися обмеження, які накладає на навчальний процес час навчання. У посібник включено найбільш важливі аспекти наукового та технічного знання, а також мінімальний набір засобів MS Access, які необхідні для створення бази даних. У виданні подано глосарій зі списком термінів, список рекомендованої літератури, а також додатки, в яких наведено теми лабораторних занять, теми індивідуальних науково-дослідних завдань (ІНДЗ) і розглянуто методику виконання ІНДЗ на прикладі туристичного бізнесу. За рамками посібника залишився матеріал, який студенти можуть самостійно освоїти в процесі практичної роботи з рекомендованою літературою та довідкою СКБД MS Access.

Посібник апробований від час викладення дисципліни «Бази даних» студентам Харківського національного університету міського господарства імені О. М. Бекетова. У ході викладення матеріалу навчального посібника використовуються **навігаційні підказки** у вигляді позначок, які допоможуть зорієнтуватися у структурі навчального посібника:

[ ] – найважливіші теоретичні відомості;

напівжирне написання – терміни програми MS Access;

курсивне написання – назви, які вводить студент.

Для визначення рівня засвоєння матеріалу кожного підрозділу пропонуються питання для самодіагностики. Засвоєння в повному обсязі матеріалу даного навчального посібника допоможе розвинути здатність до подальшого навчання, самостійного розвитку та оволодіти інструментальними засобами для створення реляційних баз даних у випадку вирішення з їх допомогою прикладних завдань геоінформаційного характеру.

РОЗДІЛ 1
ЗАГАЛЬНІ ВІДОМОСТІ ПРО БАЗИ ДАНИХ

У даному розділі розглядаються: концепція побудови баз даних, основні теоретичні поняття баз даних, системи керування базами даних (СКБД), вимоги до бази даних та СКБД, види моделей даних та їх призначення, склад реляційної та інфологічної моделей даних.

1.1 Концепція побудови баз даних

Об'єкт, предмет і методологія баз даних

Історія розвитку баз даних

Історія розвитку баз даних нараховує більше 40 років. Можна виділити такі етапи розвитку баз даних: використання файлів і файлових систем, перехід до концепції баз даних, використання баз даних на персональних комп'ютерах і створення розподілених баз даних.

Файли і файлові системи

В історії обчислювальної техніки можна прослідкувати розвиток двох основних областей її використання. Перша область – застосування електронно-обчислювальних машин (ЕОМ) для виконання чисельних розрахунків. Характерною особливістю даної сфери використання ЕОМ є наявність складних алгоритмів обробки, які застосовуються до простих за структурою даних, об'єм яких порівняно невеликий. Друга область – використання ЕОМ в автоматичних або автоматизованих інформаційних системах. **Інформаційна система** є програмно-апаратним комплексом, що забезпечує виконання таких функцій:

- надійне зберігання інформації в пам'яті комп'ютера;
- виконання специфічних для даного застосування перетворень інформації й обчислень;
- надання користувачам зручного і легко освоюваного інтерфейсу.

Інформаційні системи мають справу з великими об'ємами інформації, яка має складну структуру. Прикладами інформаційних систем є банківські системи, автоматизовані системи управління підприємствами, системи резервування авіаційних або залізничних квитків.

У традиційних пакетних системах обробки інформації, які функціонували донедавна на машинах типу ЄС ЕОМ, дані були організовані у ви-

гляді непов'язаних між собою локальних інформаційних файлів лінійної структури. Суть такого підходу до організації інформаційного забезпечення полягає в тому, що файли проектуються окремо для кожної конкретної задачі чи для їх комплексів. Такі системи називаються файловими.

Незважаючи на відносну простоту організації, файлові системи мають ряд недоліків, головними з яких є такі.

1. **Надлишковість даних**. Файлові системи мають значну надлишковість, оскільки нерідко для розв'язування різних задач управління використовуються одні й ті самі дані, розміщені в різних файлах. Через дублювання даних у різних файлах зовнішня пам'ять використовується неекономно, інформація одного й того самого об'єкта управління розподіляється між багатьма файлами. При цьому досить важко уявити загальну інформаційну модель предметної області.

2. **Неузгодженість даних**. Ураховуючи, що одна й та сама інформація може розміщуватися в різних файлах, технологічно важко простежити за внесенням змін одночасно в усі файли. Через це може виникнути неузгодженість даних, коли одне й те саме поле в різних файлах може мати різні значення.

3. **Залежність структур даних і прикладних програм**. При файловій організації логічна і фізична структури файла мають відповідати їх опису в прикладній програмі. Прикладна програма повинна бути модифікована при будь-якій зміні логічної чи фізичної структури файла. Оскільки зміни в одній програмі часто потребують внесення змін до інших інформаційно пов'язаних програм, то інколи простіше створити нову програму, ніж вносити зміни в стару. Тому цей недолік файлових систем призводить до значного збільшення вартості супроводження програмних засобів. Зазначимо, що інколи вартість супроводження програмних засобів може досягати близько 70 % від вартості їх розробки.

Розглянуті недоліки послужили тим поштовхом, який змусив розробників інформаційних систем запропонувати новий підхід до управління інформацією. Таким підходом стала концепція баз даних.

Основні підходи до побудови баз даних

Складовими частинами концепції баз даних є сукупність принципів і методів проектування та використання баз даних. У рамках цієї концепції для зберігання інформації про предметну область застосовуються: база даних – пойменована і структурована сукупність взаємопов'язаних даних і

нова програмна система – система керування базами даних (СКБД).

У 1963 році С. Бахманом була побудована перша промислова база даних IDS із мережною моделлю даних, яка все ще характеризувалася надлишковістю даних та їх використанням тільки для однієї програми. Доступ до даних здійснювався за допомогою відповідного програмного забезпечення. У 1969 році сформувалася група, яка створила набір стандартів CODASYL для мережної моделі даних. Фактично почала використовуватися сучасна архітектура баз даних.

Суттєвий стрибок у розвитку технології баз даних дала парадигма реляційної моделі даних, яку запропонував Кодд Е. Ф. у 1970 році. Під **парадигмою** розуміється наукова теорія, втілена в систему понять, що відображають суттєві риси дійсності. Тепер логічні структури могли бути отримані з одних і тих же фізичних даних, тобто доступ до одних і тих же фізичних даних міг здійснюватися різними додатками та різними шляхами. Стало можливим забезпечення цілісності і незалежності даних.

Бази даних на персональних комп'ютерах

Персональні комп'ютери (ПК) нестримно увірвалися в наше життя і буквально перевернули наше уявлення про місце та роль обчислювальної техніки в житті суспільства. Комп'ютери стали ближчими кожному користувачеві. Вони мають безліч програм, призначених для роботи непідготовлених користувачів. Програми ПК (редактори текстів, електронні таблиці та інші) стали простими у використанні й інтуїтивно зрозумілими.

З'явилися системи керування базами даних, які дозволяли створювати бази даних для зберігання значних об'ємів інформації. Вони мали зручний інтерфейс для заповнення даних та вбудовані засоби для генерації різних звітів. Ці програми дозволяють автоматизувати багато облікових функцій, які раніше велися вручну. Прикладом використання інформаційних систем може бути інформаційна система будь-якого супермаркету.

Ці бази даних мають такі особливості:

- СКБД розраховані на створення баз даних в основному з монопольним доступом, оскільки комп'ютер не був приєднаний до мережі і база даних на ньому створювалася для роботи одного користувача.
- Більшість СКБД мають розвинений і зручний для користувача інтерфейс, а також інструментарій для розробки готових додатків без програмування. Інструментарій включає готові елементи додатка у вигляді шаблонів екранних форм, звітів та графічних конструкторів запитів.

- У настільних СКБД підтримувалися як високорівневі мови маніпулювання даними типу SQL, так і низькорівневі мови маніпулювання даними на рівні окремих рядків таблиць.

Розподілені бази даних

Відомо, що історія розвивається за спіраллю. Після процесу «персоналізації» почався зворотний процес – інтеграція. Створюються локальні мережі, і все більше інформації передається між комп'ютерами. Гостро постає завдання узгодженості даних, що зберігаються й обробляються в різних місцях, але логічно пов'язані один з одним. Виникають завдання, пов'язані з паралельною обробкою транзакцій – послідовностей операцій над базою даних, що переводять її з одного несуперечливого стану в інший несуперечливий стан. Вирішення цих завдань приводить до появи розподілених баз даних, які зберігають усі переваги настільних СКБД, дозволяють організувати паралельну обробку інформації і підтримку цілісності баз даних.

Більшість сучасних СКБД можуть працювати на комп'ютерах із різною архітектурою і під різними операційними системами. При цьому для користувачів доступ до даних, керованих СКБД на різних платформах, практично невиразний. Необхідність підтримки багатокористувальницької роботи з базою даних і можливість децентралізованого зберігання даних зажадали розвитку засобів адміністрування баз даних із реалізацією загальної концепції засобів захисту даних.

До цього етапу можна також віднести розробку стандартів з мов опису та маніпулювання даними, технологій з обміну даними між різними СКБД, наприклад, протокол ODBC (Open DataBase Connectivity) фірми Microsoft. Прикладами СКБД, які відносяться до даного етапу, є MS Access і всі сучасні сервери баз даних Oracle, MS SQL, SQL Base та інші сучасні сервери баз даних, яких зараз налічується декілька десятків.

Перспективи розвитку систем керування базами даних

Цей етап характеризується появою нової технології доступу до даних – Інтернету. Для роботи з видаленою базою даних використовується стандартний браузер Інтернету, наприклад, Microsoft Internet Explorer, і для кінцевого користувача процес звернення до даних відбувається аналогічно пошуку у Всесвітній павутині. При цьому вбудований у завантажувану користувачем HTML-сторінку код, який написаний на мові Java, Java-Script, Perl і інших, відстежує всі дії користувача й транслює їх у SQL-запити до

бази даних. Зручність даного підходу привела до того, що він почав використовуватися не тільки для видаленого доступу до баз даних, а й для користувачів локальної мережі підприємства. Прості завдання обробки даних, які не пов'язані зі складними алгоритмами, досить просто й ефективно можуть бути побудовані за даною архітектурою. Для підключення нового користувача до можливості використання даного завдання не потрібна установка додаткового клієнтського програмного забезпечення. Проте алгоритмічно складні завдання рекомендується реалізовувати в архітектурі «клієнт-сервер» із розробкою спеціального клієнтського програмного забезпечення.

У кожного з вищеперелічених підходів до роботи з даними є свої переваги і недоліки, які й визначають сферу застосування того або іншого методу. У даний час усі підходи досить широко використовуються.

Таким чином, використання концепції баз даних дозволяє забезпечити: надійність, простоту й легкість використання та збереження даних, стандартизувати дані в межах однієї предметної області, а також скоротити дублювання інформації шляхом структуризації даних.

Основні поняття баз даних, терміни і визначення

Визначимо основні поняття баз даних, а саме: інформаційна система, об'єкт, предмет, предметна область, дані, модель даних, база даних, система керування базами даних. Інші поняття розглянемо в процесі викладення навчального матеріалу.

Інформаційна система – це програмно-апаратний комплекс, функції якого полягають у надійному збереженні інформації, наданні користувачу зручного інтерфейсу та виконанні специфічних операцій з перетворення та пошуку необхідної інформації.

Об'єкт – це явище зовнішнього світу. Це або щось реально існуюче – людина, товар, виріб, або процес – облік народжуваності, отримання товарів, випуск виробів.

Властивість об'єкта. Кожен об'єкт має певні властивості. Наприклад, об'єкт «Людина» володіє такими властивостями: зріст, ім'я, дата народження, а об'єкт «Виріб» – такими властивостями: якість, дата виготовлення, зовнішній вигляд.

Зв'язки між об'єктами. Існує безліч зв'язків між об'єктами. Наприклад: **людина** купує, продає, створює **виріб**; **виріб** створюється, купується, продається **людиною**. Серед усієї множини властивостей і взаємозв'язків між об'єк-

тами має сенс виділяти лише істотні, важливі з погляду споживача інформації.

Предмет – це модель реального об'єкта, в якому зафіксовані лише виділені для інформаційної системи властивості і зв'язки. Сукупність відібраних предметів утворює об'єктне ядро предметної області, а сукупність їх взаємозв'язків – структуру фрагмента дійсності.

Предметна область є абстрактною картиною реальної дійсності, певна частина якої фіксується як модель фрагмента дійсності. Поняття «Предметна область» відповідає точці зору споживача на об'єктне ядро. Виділяються тільки ті об'єкти, властивості об'єктів та зв'язки між об'єктами, які мають певну цінність і повинні бути збережені в базі даних. У кожен момент часу предметна область знаходиться в одному зі станів, який характеризується сукупністю об'єктів і їх взаємозв'язків. З часом одні об'єкти зникають, інші з'являються, змінюються властивості і взаємозв'язки. Проте нові стани, що виникають, вважаються станами тієї ж предметної області. Таким чином, предметну область доцільно розглядати як систему, що складається з певної послідовності станів.

Дані – це інформація, подана у формалізованому вигляді, придатному для пересилання, інтерпретування чи оброблення за участю людини або автоматичними засобами.

Модель даних – це інтегрований набір понять для опису даних, зв'язків між ними і обмежень, що накладаються на дані. На основі моделі даних будується база даних.

База даних – це пойменована і структурована сукупність взаємопов'язаних даних, що відображає стан об'єктів та їх відносин у даній предметній області. База даних становить інтегроване сховище даних, яке використовується багатьма споживачами і забезпечує незалежність даних від прикладних програм.

Архітектура бази даних – це різновид (узагальнення) структури, в якій будь-який елемент може бути замінений на інший елемент, характеристики входів і виходів якого ідентичні першому елементу.

Система керування базами даних (СКБД) – це комплекс програмних і мовних засобів загального та спеціального призначення, які необхідні для створення бази даних, підтримування її в актуальному стані, маніпулювання даними й організації доступу до них різних користувачів в умовах прийнятої технології обробки даних. Система керування базами даних виступає посередником між користувачем і базою даних.

Принцип організації зв'язку кінцевих користувачів і прикладних програм з базою даних показано на рис. 1.1.

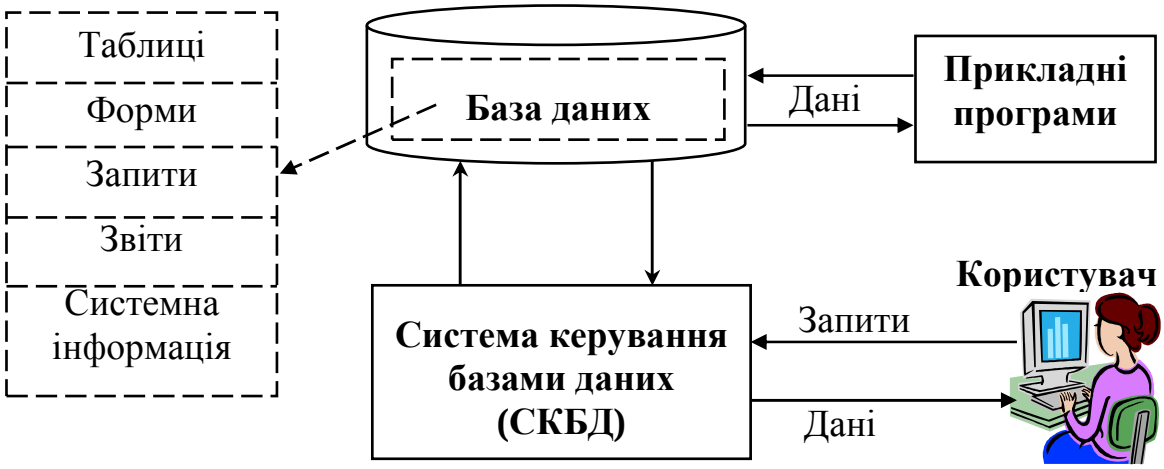


Рис. 1.1 – Взаємодія користувача і прикладних програм із базою даних

Вирішення інформаційних завдань зводиться в основному до пошуку необхідної інформації в базі даних за запитами користувачів. У запитах зазначаються умови пошуку та відображення інформації про конкретні об'єкти, наприклад, про наявність залишків товару на кінець залікового періоду роботи магазину в разі аналізу господарської діяльності.

Методологія та архітектура баз даних

Методологія – це сукупність методів вирішення проблеми. Методологія бази даних визначається в процедурі проектування, але проявляється і в процедурі використання.

Методологія проектування баз даних – це єдина сукупність правил, засобів і процедур, які дають можливість скоротити час і поліпшити якість проектування. Існує багато різновидів методології розгляду баз даних у класичному підході. Однак найчастіше дотримуються методології ANSI/SPARC, згідно з якою архітектура бази даних має три рівні. На рис. 1.2 показана трирівнева архітектура та етапи проектування бази даних.

На етапі словесного опису бази даних (БД) здійснюється формулювання й аналіз вимог, встановлюються цілі організації та визначаються вимоги до бази даних. Вони складаються з вимог користувачів і специфічних вимог прикладних програм. Усі вимоги документуються у формі, доступній кінцевому користувачу і проектувальнику бази даних.

Перший етап проектування відповідає концептуальному рівню бази даних. На цьому етапі формується інформаційно-логічна (інфологічна) мо-

дель бази даних, у якій подається семантика предметної області. Результатом цього етапу є загальне та наочне подання інформаційних вимог користувачів.

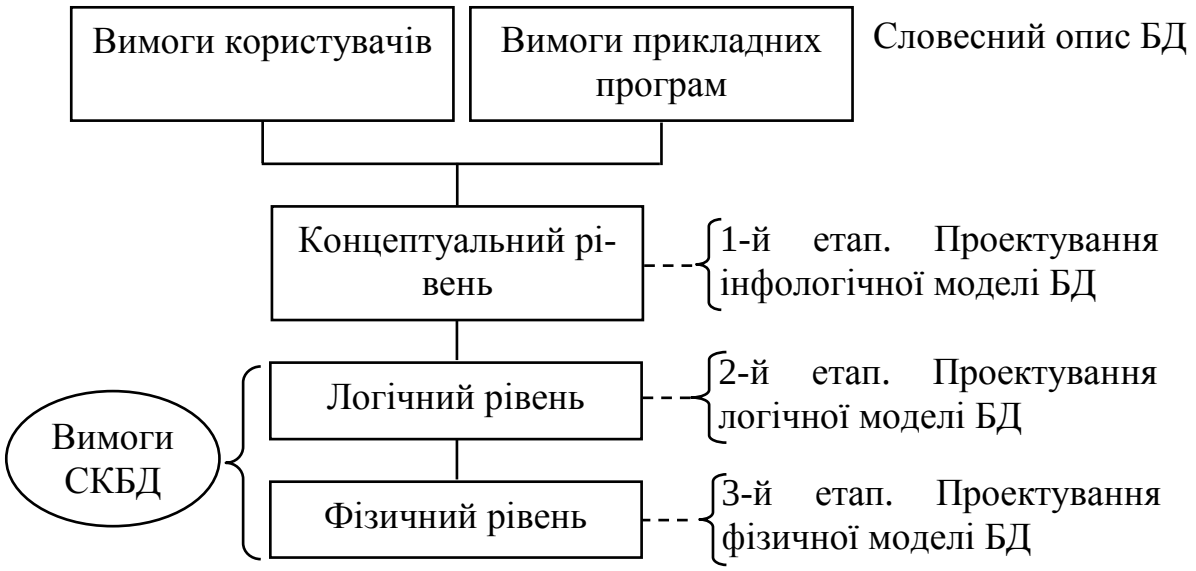


Рис. 1.2 – Трирівнева архітектура баз даних

Другий етап проектування відповідає логічному рівню бази даних. У процесі логічного проектування здійснюється перетворення інфологічної моделі бази даних у логічну модель, яка враховує особливості використовуваної СКБД. Основною метою етапу є усунення надмірності даних із застосуванням спеціальних правил нормалізації. Мета нормалізації – мінімізувати повторення даних і можливі структурні зміни бази даних при процедурах оновлення. Отримана логічна структура бази даних може бути оцінена кількісно за допомогою різних характеристик (число звернень до логічних записів, об'єм даних у кожному додатку, загальний об'єм даних). На основі цих оцінок логічна структура може бути вдосконалена з метою досягнення більшої ефективності.

На етапі фізичного проектування вирішуються питання, що пов'язані з продуктивністю системи, визначаються структури зберігання даних та методи доступу. На цьому етапі здійснюється тестування бази даних з урахуванням вимог кінцевих користувачів.

Процедура управління базою даних є найбільш простою в однокористувальницькому режимі. У багатокористувальницькому режимі і в розподілених базах даних процедура значно ускладнюється. При одночасному доступі декількох користувачів без застосування спеціальних заходів мож-

ливе порушення цілісності. Для усунення цього явища використовують систему транзакцій і режим блокування таблиць або окремих записів. **Транзакція** – це процес зміни файла, запису або бази даних, викликаний передачею одного вхідного повідомлення.

Процес проектування є тривалим і трудомістким. Основними ресурсами проектувальника бази даних є його власна інтуїція та досвід, тому якість рішення може виявитися низькою. Основними причинами низької ефективності проектованих баз даних можуть бути:

- недостатньо глибокий аналіз вимог (початковий етап проектування), включаючи їх семантику і взаємозв'язок даних;
- велика тривалість процесу структуризації, що робить цей процес утомливим і важко виконуваним при ручній обробці.

У цих умовах важливого значення набувають питання автоматизації розробки.

Методологія використання баз даних

База даних використовується зазвичай не самостійно, а є складовою частиною різних інформаційних систем: банків даних, інформаційно-пошукових і експертних систем, систем автоматизованого проектування, автоматизованих робочих місць, автоматизованих систем управління.

У базі даних є три рівні подання бази даних (див. рис.1.2): концептуальний, логічний і фізичний. У процедурі використання найчастіше мають справу з логічною моделлю і значно рідше з концептуальною та фізичною моделями.

Узагальнене подання всіх трьох рівнів містить словник даних. Словник даних є як би внутрішньою базою даних, що містить централізовані відомості про всі типи даних, їх імена, структуру, а також інформацію про їх використання. Перевагою словника даних є ефективне накопичення та управління інформаційними ресурсами предметної області. Його застосування дозволяє зменшити надмірність і суперечність даних при їх введенні, здійснити просте та ефективне управління при їх модифікації, спростити процедуру проектування бази даних за рахунок централізації управління даними, встановити зв'язки з іншим користувачами.

У процесі використання баз даних є операції оновлення (запис, видалення, модифікація даних) і запит-відповідь (читання).

У загальному випадку процес запиту складається з ряду етапів. Користувач повинен знати структуру бази даних або звернутися до адмініст-

ратора бази даних. На першому етапі користувач має з'ясувати, які форми документів йому потрібні. Це можуть бути не тільки логічні моделі користувача, але й різні їх модифікації при різних поєднаннях полів. Оскільки логічні, а тим більше модифіковані логічні, моделі можуть відрізнятися від логічної моделі бази даних, слід визначити, які поєднання полів необхідні для машинних документів, що виводяться. Ці поєднання утворюються за допомогою елементарних правил, що вивчаються реляційною алгеброю і реляційним численням. Далі правила слід трансформувати у відповідні варіанти звернення до СКБД через її інтерфейс. Це можуть бути меню, екранні форми, мова запитів, запит за прикладом, режим перегляду таблиць бази даних. Результат може бути поданий у вигляді таблиць або звітів.

При експлуатації бази даних використовують дві специфічні операції: навігацію і специфікацію.

Навігація – це операція, результат якої представлений єдиним об'єктом, отриманим при проходженні шляху по логічній структурі (сукупності таблиць, зв'язок між якими здійснюється за ключами) бази даних.

Специфікація – це операція, результатом якої є нова структура (таблиця), побудована на основі структур таблиць бази даних. Вона має назву «вигляд».

Поняття схеми бази даних, стандарт SPARC

Схема бази даних – це загальний опис бази даних. Існують різні типи схем бази даних, які визначаються відповідно до рівнів абстракції багаторівневої архітектури, показаної на рис. 1.2. На кожному рівні використовується своя мова опису схеми бази даних.

Зовнішній рівень є словесним описом вхідних і вихідних повідомлень, а також даних, які доцільно зберігати в базі даних. При проектуванні бази даних необхідно на зовнішньому рівні вивчити функціонування об'єкта керування, для якого проектується база даних, усю первинну та вихідну документацію для визначення даних, які потрібно зберігати в базі даних, а також вимоги користувачів і прикладних програмістів. На цьому рівні не виключена наявність елементів дублювання, надлишковості та неузгодженості даних. Для усунення цих аномалій і протиріч зовнішнього опису даних виконується інфологічне та логічне проектування.

Інфологічний рівень становить інформаційно-логічну модель предметної області, в якій виключено надлишковість даних і відображено інформаційні особливості об'єкта керування без урахування особливостей і

специфіки конкретної СКБД. Інфологічне подання даних орієнтоване переважно на людину, яка проектує чи використовує базу даних. На етапі проектування інфологічної моделі будується концептуальна схема, яка є структурованою моделлю бази даних без урахування особливостей СКБД. Для розробки концептуальної схеми широко застосовується метод «сутність-зв'язок».

Логічний рівень ураховує специфіку й особливості конкретної СКБД. Цей рівень подання даних орієнтований більше на комп'ютерну обробку і на програмістів, які займаються її розробкою. Проектування логічної моделі здійснюється шляхом перетворення інфологічної моделі даних у логічну модель. Результат цього етапу – логічна схема бази даних у вигляді взаємопов'язаних таблиць. Основною метою етапу є усунення надмірності даних із використанням спеціальних правил нормалізації та мінімізація можливих структурних змін бази даних.

На фізичному рівні вирішуються питання, що пов'язані з продуктивністю системи, створюються таблиці та інші об'єкти бази даних і здійснюється тестування бази даних з урахуванням вимог кінцевих користувачів. Результатом цього етапу є схема бази даних у вигляді взаємопов'язаних таблиць.

Значення баз даних у побудові інформаційних систем

У наш час – в епоху інформаційного вибуху – розроблюється і впроваджується велика кількість найрізноманітніших автоматизованих інформаційних систем (АІС) з дуже широким спектром використання.

У науковій літературі існує досить значне розмаїття класифікацій інформаційних систем (ІС). Різні автори залежно від своїх завдань та точок зору виділяють ті чи інші критерії і розподіляють пріоритети між ними. Зупинимось на одному з таких підходів, який, на погляд авторів, найбільше узгоджується з іншими темами цієї дисципліни. Отже, ІС класифікуються:

- за призначенням – фактографічні, документальні і змішані;
- за мовами – замкнуті системи, системи з базовою мовою та змішані;
- за локалізацією – локальні й розподілені;
- за структурами даних – ієрархічні, мережного типу та реляційні.

Розглянемо кожний із критеріїв.

Призначення ІС. Документальні системи зорієнтовані на обробку та зберігання документа (порівняно великої за розміром послідовності симво-

лів), внутрішню структуру якого система (майже) повністю ігнорує, тобто він неподільний (атомарний) з точки зору системи. Споживачем результатів пошуку виступає, як правило, кінцевий користувач.

Фактографічні системи оперують фактами (даними) різних типів, що пов'язані в системі в більш чи менш складні структури. Дані, що є результатом пошуку, можуть стати складовою частиною звітів або використовуються різноманітними обчислювальними процесами.

Змішані системи включають у себе в тих чи інших пропорціях риси обох вищеназваних варіантів. Переважну більшість сучасних систем для персональних комп'ютерів слід віднести до категорії змішаних.

Мови ІС. Системи з базовою мовою передбачають взаємодію користувача із СКБД із середовища якоїсь іншої мови програмування, де і виконується більшість постпошукових перетворень даних. Такий підхід зручний для розробки різного роду систем як надбудов над СКБД, бо дає можливість створювати високоефективні програми постпошукової обробки даних. Замкнуті системи самостійно забезпечують користувача всіма необхідними засобами як для локалізації даних, так і для їх постпошукової чи передпошукової обробки. Недоліком таких систем є те, що в них відсутні (або малоефективні) засоби для розробки надбудов – проблемно-орієнтованих комплексів.

Змішані системи передбачають наявність обох можливостей двох попередніх підходів і є найбільш поширеними на сьогодні.

Локалізація ІС. Локальність передбачає розташування всього програмного забезпечення і даних на одному ізольованому комп'ютері, а розподіленість означає розташування системи на мережі комп'ютерів із певною стратегією рознесення даних.

Структури даних. Структури даних, що підтримуються в системі бази даних, – це важливий фактор, який впливає як на виразову потужність, так і на ефективність функціонування.

Для систем з ієрархічною структурою базовою структурою даних є дерево. Як правило, вони мають найвищу ефективність функціонування, але виразові можливості їх відносно низькі.

Системи із структурами даних типу «мережа» мають значно кращі виразові можливості, але дещо програють у ефективності функціонування, точніше, від користувача вимагається значно вищий рівень кваліфікації для ефективної експлуатації таких систем.

В останні десятиріччя найбільшого розповсюдження, особливо для персональних комп’ютерів, зазнали СКБД реляційного типу, для яких характерна щонайпростіша структура даних (плоский файл). Але одночасно суттєво підвищується рівень мов маніпулювання даними, що максимально употужнює виразові можливості та знижує ефективність функціонування. Для таких систем потрібні потужні комп'ютери, і вони значно чутливіші (порівняно з попередниками) до зростання об'ємів даних.

Система керування базами даних як спеціальний програмний комплекс

Склад і функції систем керування базами даних

Усі функції СКБД можна об'єднати в такі групи:

1. **Керування даними**. Задачами керування даних є підготовка даних та їх контроль, занесення даних до бази, структуризація даних, забезпечення цілісності та безпеки даних.
2. **Доступ до даних**. Пошук і селекція даних, перетворення їх у форму, зручну для подальшого використання.
3. **Організація і ведення зв'язку з користувачем**. Ведення діалогу, видача діагностичних повідомлень про помилки в роботі з базою даних.

Для опису даних, організації спілкування та виконання процедур пошуку й різних перетворень із даними застосовуються мовні засоби СКБД. У склад мовних засобів входять:

- мова опису даних;
- мова маніпулювання даними, що забезпечує виконання основних операцій над даними – введення, модифікацію і вибірку;
- мова структурованих запитів SQL (Structured Query Language), що забезпечує управління структурою бази даних і маніпулювання даними, а також є стандартним засобом доступу до видалених баз даних;
- мова запитів за зразком QBE (Query By Example), що забезпечує візуальне конструювання запитів до бази даних.

Мова опису даних призначена для опису даних на різних рівнях абстракції. Опис усіх імен, типів і розмірів полів зберігається в пам'яті разом із даними.

Мова маніпулювання даними може бути базовою чи автономною. Базова мова – це одна з традиційних мов програмування (БЕЙСІК, С та ін.). Системи, що використовують базову мову, називають відкритими. Авто-

номна мова маніпулювання даними – це власна мова СКБД, яка дає змогу виконувати різні операції з даними. У сучасних СКБД для спрощення процедур пошуку даних у базі даних передбачено такі мови запитів, як SQL і QBE. Мова запитів QBE – це реалізація запитів за зразком у вигляді таблиць. Для визначення запиту до бази даних користувач повинен заповнити надану системою таблицю QBE і визначити в ній критерії пошуку та вибору даних.

Також було розроблено стандартний інтерфейс мови CLI (Common Language Interface) для всіх основних варіантів мови SQL. Цей інтерфейс був формалізований фірмою Microsoft і дістав назву ODBC (Open Database Connectiviti – відкритий доступ до даних). ODBC – це інтерфейс доступу до даних засобами СКБД. Він має цілий набір драйверів, за допомогою яких одна СКБД може працювати з даними інших систем.

Види систем керування базами даних

За характером застосування розрізняють персональні та багатокористувальницькі СКБД.

Персональні СКБД забезпечують можливість створення локальних баз даних, що працюють на одному комп'ютері. До персональних СКБД відносяться MS Access, Paradox, FoxPro та ін. Більшість персональних СКБД мають розвинений і зручний для користувача інтерфейс, а також інструментарій для розробки готових додатків без програмування. Інструментарій включає готові елементи додатка у вигляді шаблонів екранних форм, звітів, графічних конструкторів запитів. У таких СКБД підтримуються як високорівневі мови маніпулювання даними типу SQL, так і низькорівневі мови маніпулювання даними на рівні рядків таблиць.

Багатокористувальницькі СКБД дозволяють створювати інформаційні системи, що функціонують в архітектурі «клієнт–сервер». Найбільш відомими СКБД є Oracle, Microsoft SQL Server, InterBase. Багатокористувальницькі СКБД зберігають усі переваги вищезгаданих СКБД та дозволяють організувати паралельну обробку інформації. Більшість сучасних СКБД можуть працювати на комп'ютерах із різною архітектурою і під різними операційними системами. Необхідність підтримки багатокористувальницької роботи з базою даних і можливість децентралізованого зберігання даних призвели до розвитку засобів адміністрування баз даних із реалізацією загальної концепції засобів захисту даних.

Вимоги до бази даних і СКБД

Проектування баз даних – це ітераційний, багатоетапний процес прийняття обґрунтованих рішень у процесі аналізу інформаційної моделі предметної області, вимог до даних з боку прикладних програмістів і користувачів, синтезу логічних і фізичних структур даних, аналізу та обґрунтування вибору програмних і апаратних засобів. Етапність проектування даних пов'язана з багаторівневою організацією даних.

Згідно з пропозиціями дослідницької групи із систем керування даними Американського національного інституту стандартів ANSI/SPARC, як правило, виділяється три рівні подання даних: зовнішній (з точки зору кінцевого користувача і прикладного програміста), концептуальний (з точки зору СКБД) та внутрішній (з точки зору системного програміста). Згідно з цією концепцією зовнішній рівень є частиною концептуальної моделі, що необхідна для реалізації якогось запиту чи прикладної програми. Модель ANSI/SPARC не стала стандартом, проте вона все ще є основою для розуміння деяких функціональних особливостей СКБД.

На базі моделі ANSI/SPARC запропоновані й інші варіанти багаторівневого подання даних, у яких під зовнішнім рівнем розуміють загальні поняття, які пов'язані з вивченням і аналізом інформаційних потоків предметної області, а для структуризації даних вводиться допоміжний рівень, який називається інфологічним, або інформаційно-логічним. Зовнішній рівень у цьому випадку виступає як окремий етап проектування, на якому вивчається все позамашинне інформаційне забезпечення: форми документування та подання даних, а також зовнішнє середовище, в якому функціонуватиме база даних з точки зору методів фіксації, збирання та передавання інформації в базу даних.

Така концепція доцільніша з точки зору розуміння процесу проектування бази даних. Тому будемо дотримуватися багаторівневого подання даних, що включає зовнішній рівень і три рівні подання даних у базі даних на етапі проектування, створення та супроводження бази даних (див. рис. 1.2).

Мета багаторівневої архітектури полягає у відокремленні призначеного для користувача подання бази даних від її фізичного уявлення. Причини, з яких бажано виконувати дане розділення, такі:

- кожен користувач повинен мати можливість звертатися до одних і тих же даних, використовуючи своє власне уявлення про них, а також змінювати своє уявлення про дані, причому ця зміна не повинна впливати

на інших користувачів;

- користувачі не повинні безпосередньо мати справи з подробицями фізичного зберігання даних у базі;
- адміністратор бази даних повинен мати можливість змінювати структуру зберігання даних у базі, не впливаючи на призначені для користувача представлення.

Види моделей даних та їх призначення

Моделі даних визначаються трьома компонентами: організацією даних, обмеженнями цілісності та багатофункціональністю припустимих операцій. У теорії систем керування базами даних виділяють моделі трьох основних типів: ієрархічну, мережну і реляційну.

Ієрархічна модель даних. Термінологічною основою для ієрархічної і мережної моделей є поняття: атрибут, агрегат та запис. Під атрибутом (елементом даних) розуміється найменша пойменована структурна одиниця даних. Пойменована множина атрибутів може утворювати агрегат даних. У деяких випадках окремо взятий агрегат може складатися з множини екземплярів однотипних даних, або, як ще говорять, бути множинним елементом. Нарешті, записом називають складовий агрегат, який не входить до складу інших агрегатів.

В ієрархічній моделі всі записи, агрегати й атрибути бази даних утворюють ієрархічно організований набір, який зручно зображувати за допомогою деревоподібного графа (рис.1.3).

В ієрархічній моделі всі елементи пов'язані відношеннями підпорядкованості, і будь-який елемент може підкорятися тільки одному іншому елементу.

Мережна модель даних. Мережний підхід до організації даних є розширенням ієрархічного. В ієрархічних структурах запис-послідовник повинен мати одного попередника; у мережній структурі даних послідовник може мати будь-яке число попередників (рис. 1.4).

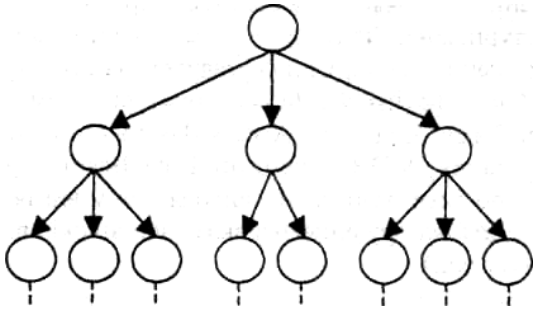


Рис. 1.3 – Схема ієрархічної моделі даних

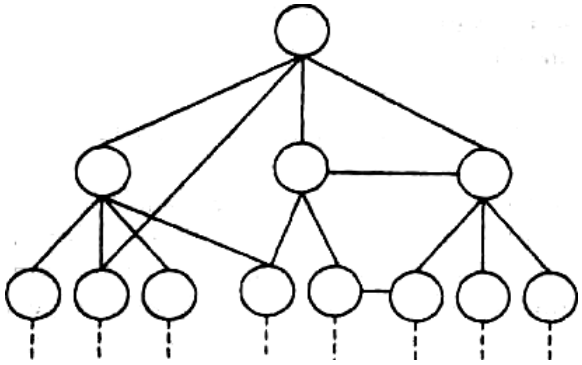


Рис. 1.4 – Схема мережної моделі даних

Серед переваг систем керування даними, заснованих на ієрархічній чи мережній моделях, можуть бути їх компактність і висока швидкодія, а серед недоліків – високий ступінь залежності від конкретних даних.

Реляційна модель даних. Реляційна модель даних вперше була запропонована британським вченим, співробітником компанії IBM Едгаром Коддом. У даний час ця модель є фактичним стандартом, на який орієнтуються практично всі сучасні СКБД. У реляційній моделі досягається набагато вищий рівень абстракції даних, ніж в ієрархічній або мережній моделях. Реляційна модель даних ґрунтується на понятті відношення, і становить собою безліч елементів, що називаються кортежами. Наочною формою подання відношення є звична для сприйняття двовимірна таблиця. Таблиця має рядки (записи) і стовпці (колонки). Кожний рядок таблиці має однакову структуру і складається з полів. Рядкам таблиці відповідають кортежі, стовпцям – атрибути відношення.

Створена Коддом теорія нормалізації відношень виявилася, по суті, єдиною формалізованою теорією, за допомогою якої можна спроектувати оптимальну логічну модель даних. Цю теорію можна використовувати не лише при проектуванні баз даних у середовищі реляційних СКБД, а й для СКБД, які підтримують інші моделі даних. Таким чином, будь-яку логічну модель спочатку проектують як нормалізовану реляційну модель, а потім відображують на ту модель, яку підтримує вибрана СКБД. Реляційна модель має найбільш широке використання для створення баз даних різного призначення, в тому числі економічного характеру.

Питання для самодіагностики

1. Дайте визначення понять «інформаційна система», «об'єкт», «предмет», «предметна область».
2. Дайте визначення поняттю «база даних».

3. Поясніть підходи до побудови баз даних.
4. Поясніть поняття схеми бази даних і стандарту SPARC.
5. Дайте визначення поняттю «система керування базами даних».
6. Дайте характеристику складу і функцій СКБД.
7. Охарактеризуйте мовні засоби СКБД. Для яких цілей вони використовуються?
8. Які вимоги ставляться до бази даних і СКБД?
9. Які види СКБД існують? Охарактеризуйте їх.
10. Які види моделей даних ви знаєте?
11. Охарактеризуйте основні моделі даних.

1.2 Реляційна модель даних

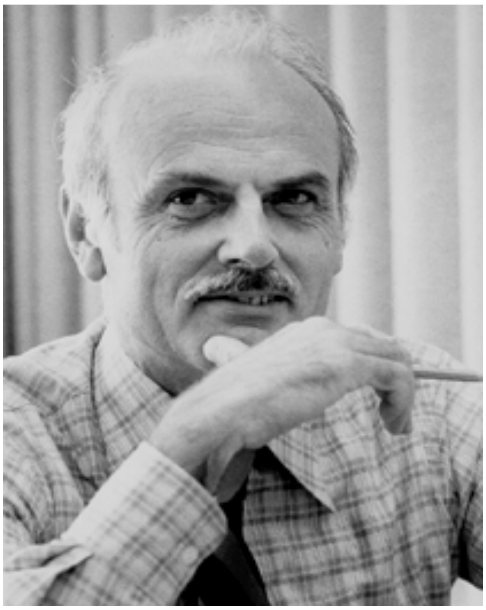


Рис. 1.5 – Едгар Кодд

Концепцію реляційної моделі даних запропонував американський вчений Кодд Едгар Франк у 1970 році (рис. 1.5)

Едгар Франк Кодд народився 23 серпня 1923 року у м. Портланд (Англія). Освіту з математики та хімії отримав у Оксфордському університеті. З 1948 року працював у США в компанії IBM як математик-програміст. Створив теорію нормалізації відношень, а в 1985 році визначив 12 правил, яким повинна відповідати будь-яка база даних, що претендує на тип реляційної. Він зробив суттєвий внесок в інші області інформатики. Помер Кодд Е. Ф. 18 квітня 2003 року.

Виникнення концепції реляційної моделі даних пов'язане з розв'язанням проблеми забезпечення незалежності даних і їх опису від прикладних програм. Створена Коддом теорія нормалізації відношень виявилася, по суті, єдиною формалізованою теорією, за допомогою якої можна спроектувати оптимальну логічну модель даних. Цю теорію можна використовувати не лише при проектуванні баз даних у середовищі реляційних СКБД, а й для СКБД, які підтримують інші моделі даних.

Таким чином, будь-яку логічну модель спочатку проектують як нормалізовану реляційну модель, а потім відображають на ту модель, яку під-

тримує вибрана СКБД.

Основні поняття та складові частини реляційної моделі даних

Основними поняттями реляційних баз даних є: відношення, атрибут, схема відношення, тип даних, домен, кортеж, таблиця, первинний ключ, зв’язок, вторинний ключ, структурна, цілісна та маніпуляційна частини моделі даних.

Значення цих понять розглянемо з використанням прикладу відношення, яке містить інформацію про викладачів кафедри (рис. 1.6).

Відношення є фундаментальним поняттям і засобом структурування даних у реляційній моделі даних. Відношенням називається будь-який зв'язок між об'єктами або їх властивостями.

Відношення має ім’я, яке задається іменником. Ім’я повинне бути унікальним у базі даних і пояснювати зміст (семантику) відношення. Засобом подання відношення в реляційній базі даних є іменована таблиця.

Атрибути і схема відношення

Атрибут – це логічно неподільна одиниця даних, яка визначає певну властивість об'єкта або зв’язок. Кожний атрибут має ім’я, певний тип (рядок символів, число та ін.), значення та інші властивості.

Ім'я атрибута задається іменником в однині. Воно повинне бути унікальним у відношенні, простим і зрозумілим.

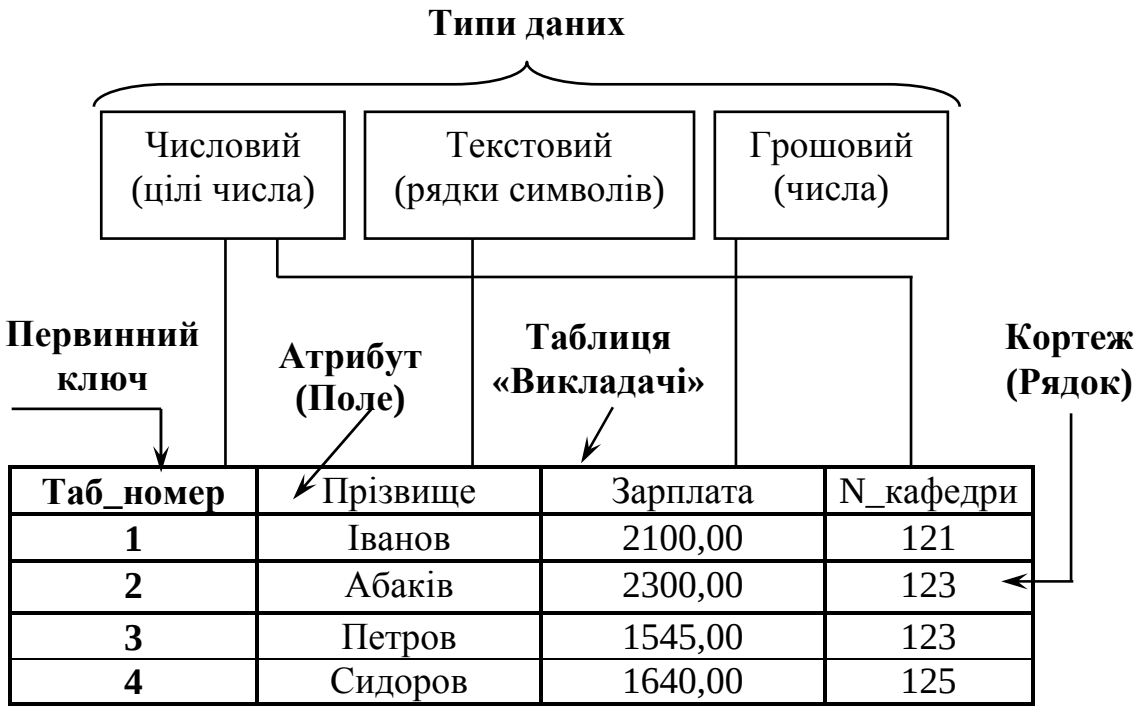
Значення атрибута – величина, що характеризує деяку властивість об'єкта або зв’язок. Значення має певний тип, формат, довжину та повинно бути атомарним (неподільним) для даної моделі. Приклади атрибутів та їх значень наведені на рис. 1.6 (первинний ключ виділено напівжирним написанням).

Структура відношення подається у вигляді схеми.

Схемою відношення є кінцева множина впорядкованих імен атрибутів – (*A*1, *A*2,..., *A*n). Опис відношення складається з його ім’я та схеми: ІМ’Я(*A*1, *A*2,..., *A*n), наприклад:

ВИКЛАДАЧ (**Таб_номер**, Прізвище, Зарплата, N_кафедри).

Для однозначної ідентифікації кортежів кожне відношення повинно мати первинний ключ. **Первинний ключ** складається з одного або декількох атрибутів. Наприклад, атрибут «**Таб_номер**» є первинним ключем відношення «ВИКЛАДАЧ» і визначає кортежі цього відношення.



Таб_номер	Прізвище	Зарплата	N_кафедри
1	Іванов	2100,00	121
2	Абаків	2300,00	123
3	Петров	1545,00	123
4	Сидоров	1640,00	125

Рис. 1.6 – Відношення з інформацією про викладачів кафедри

Об'єктні та зв'язкові відношення

Відношення можна поділити на два класи: об'єктні й зв'язкові. Об'єктні відношення зберігають дані про інформаційні об'єкти предметної області. Прикладом об’єктного відношення є відношення «ВИКЛАДАЧ». В об'єктному відношенні не повинно бути рядків з однаковим ключем.

Зв'язкове відношення містить ключі двох або більше об'єктних відношень. За цими ключами зв’язкове відношення встановлює зв'язки між відповідними об’єктними відношеннями. Наприклад, розглянемо два об'єктних відношення:

СТУДЕНТ(**Код_студента**, Прізвище);

ДИСЦИПЛІНА(**Код_дисципліни**, Назва).

Тоді відношення

ЕКЗАМЕН(**Код студента, Код дисципліни**, Оцінка)

буде зв'язковим між об'єктними відношеннями «СТУДЕНТ» і «ДИСЦИПЛІНА».

У зв'язковому відношенні допускається дублювання значень ключових атрибутів. Крім ключових атрибутів, у зв'язковому відношенні можуть бути й інші атрибути, наприклад, атрибут «Оцінка».

Тип даних, домен і кортеж

Значення даних, що зберігаються в реляційній базі даних, мають пев-

ний тип, наприклад, текстовий, числовий, грошовий, логічний, дата/час та ін. Наприклад, у табл. 1.1 наведені типи даних відношення «ВИКЛАДАЧ».

Тип даних визначає: множину значень, набір операцій, що можуть бути застосовані до значень заданого типу, та спосіб зовнішнього подання значень. Наприклад, числовий тип визначає множину чисел, над якими допустимі арифметичні операції, а зовнішнє подання – числа.

У реляційній моделі використовуються прості (скалярні) дані, значення яких є атомарними (неподільними). Це означає, що в реляційних операціях не повинна враховуватися внутрішня структура даних. Наприклад, простими даними є дані числового типу та рядки символів, оскільки в операціях вони використовуються як єдине ціле.

Таблиця 1.1 – Типи атрибутів відношення «ВИКЛАДАЧ»

Ім’я атрибута	**Тип**
Таб_номер	Цілий
Прізвище	Текстовий (20)
Зарплата	Грошовий
№_кафедри	Цілий

Якщо розглядати масив як єдине ціле і не використовувати операції над його елементами, то масив також можна вважати простим типом даних. Більш того, допустимо створити свій, скільки завгодно складний тип даних, описати можливі дії з ним, і якщо в операціях не потрібне знання внутрішньої структури даних, то такий тип даних також буде простим з погляду реляційної теорії. Наприклад, атрибут «Прізвище» (див. рис. 1.6), який має текстовий тип і рядки символів, що виражають прізвища викладачів, є атомарними. Якщо рядок розкласти в масив символів, то при переході на такий низький рівень буде втрачено сенс цього рядка як єдиного цілого, тобто це буде вже не прізвище, наприклад, Іванов.

Доменом є підмножина значень простого типу даних, які мають певний сенс і є допустимими для атрибута, що визначений на домені. Домен має унікальне ім'я в межах бази даних. Зазвичай ім’ям домену є ім’я атрибуту, але можливе й інше ім’я (рис. 1.6). Домен може бути визначений як на простому типі даних, так і на іншому домені. Для опису підмножини допустимих даних домену використовуються логічні умови. Наприклад, домен *ВВ*, що має сенс «вік викладача», можна описати як підмножину *n*

множини *N* натуральних чисел: ($BC = n \in N$: n > 18 і n < 60). А домен «Прізвища», який задано текстовим типом атрибуту «Прізвище», задає обмеження на довжину рядків – 20 символів (див. табл. 1.1).

Основне призначення доменів полягає в тому, що вони обмежують порівняння. Некоректно, з логічної точки зору, порівнювати значення з різних доменів, навіть якщо вони мають однаковий тип. Наприклад, значення різних доменів «Номери перепусток» і «Номери відділів» не є порівнянними, хоча домени визначені одним типом цілих чисел. Таким чином, поняття домену допомагає правильно моделювати предметну область.

Кортеж становить набір іменованих значень заданого типу. Іменами значень кортежу є імена атрибутів, а значеннями – припустимі значення домену даного атрибута. Відношення складається з множини кортежів.

Таблиці. Первинні ключі таблиць

Таблиці є основною формою подання відношень і основним засобом зберігання даних у реляційній базі даних. Кожному відношенню відповідає таблиця з таким же ім'ям, але, на відміну від відношення, ім'я таблиці є іменником у множині, наприклад, «Викладачі». У реляційній базі даних імена таблиць повинні бути унікальними.

Таблиця складається із заголовку, стовпців і рядків.

Стовпці таблиці називають також **полями**. На перетині стовпців і рядків містяться значення даних (див. рис. 1.6). Стовпець таблиці відповідає атрибуту відношення, має ім'я та містить дані одного з припустимих типів, наприклад, текстового, числового або ін. В одній таблиці стовпці повинні мати унікальні імена, а в різних таблицях імена стовпців можуть бути однакові, наприклад, кожна з таблиць «Викладачі» і «Студенти» може мати поле з ім'ям «Прізвище». Кількість стовпців у таблиці фіксована, список їх імен складає заголовок таблиці. При створенні таблиці її стовпці упорядковуються зліва направо у послідовності введення імен стовпців. Максимально допустиме число стовпців у таблиці, як правило, не вказується. Однак майже у всіх комерційних СКБД ця межа існує та складає приблизно 255 стовпців. У будь-якій таблиці завжди є, як мінімум, один стовпець.

Рядки таблиці відповідають кортежам відношення. На відміну від стовпців, рядки не мають імен, кількість їх не обмежена, а порядок розміщення довільний. Припустиме існування таблиці з нульовою кількістю рядків, яка називається порожньою. Порожня таблиця зберігає структуру, ви-

значену її стовпцями, але в ній не містяться дані. Стандарти реляційних баз даних не накладають обмежень на кількість рядків у таблиці, і в багатьох СКБД розмір таблиць обмежений лише вільним дисковим простором комп'ютера.

Таблиці розміщуються у файлі бази даних. Кожен рядок таблиці відповідає запису файла, тому рядки таблиці називають також записами.

Первинні ключі ідентифікують рядки оскільки рядки таблиці не мають імен. Первинний ключ становить комбінацію атрибутів таблиці, дані яких однозначно визначають кожен рядок таблиці. Якщо ключ складається з одного атрибуту, він називається простим ключем, а якщо з декількох – складеним ключем. Атрибути, з яких складається ключ, називають ключовими. Первинний ключ таблиці будемо визначати жирним шрифтом, як у СКБД Access, наприклад, **Таб_номер**.

Логічні зв'язки між таблицями. Вторинні ключі

Реляційна база даних складається з набору взаємопов’язаних таблиць. Зв’язки об’єднують інформацію окремих таблиць в єдину базу даних.

Зв’язком будемо називати спосіб, за допомогою якого дані однієї таблиці об’єднується з даними другої таблиці. Умовою об’єднання даних є збігання значень певних полів таблиць, що пов’язані між собою. Зв'язок між двома таблицями визначає їх підпорядкованість – одна з таблиць є головною, а друга – підлеглою. У головній таблиці атрибутом зв'язку є первинний ключ, а в підлеглій таблиці – вторинний, або зовнішній, ключ.

Вторинний ключ – це додаткове поле, що включається в підлеглу таблицю для встановлення зв'язку з головною таблицею. Первинний та вторинний ключі, що пов'язують таблиці, повинні мати однаковий тип. Імена цих ключів можуть бути різними.

Існують такі види зв'язків: «один-до-одного» (1:1), «один-до-багатьох» (1:M), «багато-до-одного» (M:1) та «багато-до-багатьох» (M:M).

Зв'язок «один-до-одного» означає, що одному запису в головній таблиці відповідає один запис у підлеглій таблиці, а кожному запису підлеглої таблиці – рівно один запис у головній. Наприклад, маємо дві таблиці «Викладачі» та «Фотографії». Якщо для кожного викладача таблиці «Викладачі» зберігається одна фотографія в таблиці «Фотографії», то між таблицями встановлюється зв’язок «один-до-одного» (рис. 1.7).

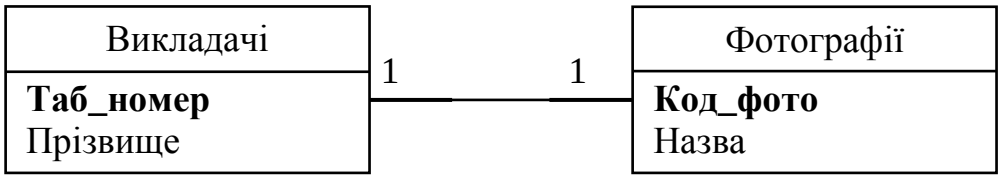


Рис. 1.7 – Організація зв’язку «один-до-одного»

Зв'язок «один-до-багатьох» означає, що одному рядку головної таблиці може відповідати будь-яка кількість рядків у підлеглій таблиці, у тому числі один рядок або жодний із рядків, а кожному рядку підлеглої таблиці – тільки один рядок у головній таблиці. Наприклад, маємо дві таблиці «Викладачі» та «Дисципліни» (рис. 1.8).

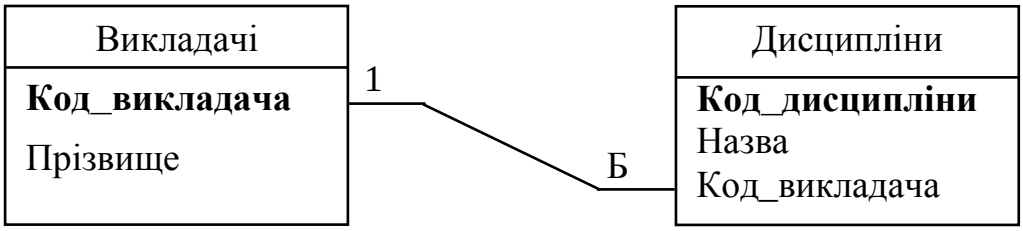


Рис. 1.8 – Організація зв’язку «один-до-багатьох»

Нехай кожний викладач може читати декілька навчальних дисциплін. Тоді у підлеглу таблицю «Дисципліни» включимо додаткове поле «Код_викладача», яке є вторинним ключем. Зв’язок між таблицями «Викладачі» та «Дисципліни» буде «один-до-багатьох». Цей вид зв'язку зустрічається найбільш часто. Зв'язок «багато-до-одного» відрізняється від зв'язку «один-до-багатьох» тільки напрямком. Якщо на зв'язок «багато-до-одного» подивитися з боку підлеглої таблиці, то вона перетворюється в зв'язок «багато-до-одного».

Зв'язок «багато-до-багатьох» має місце, коли одному рядку головної таблиці може відповідати кілька рядків підлеглої таблиці, й одночасно одному рядку підлеглої таблиці – кілька рядків головної таблиці. Наприклад, викладач ВНЗ може читати декілька навчальних дисциплін, а одну дисципліну можуть читати декілька викладачів. Тоді між таблицями «Викладачі» та «Дисципліни» треба встановити зв’язок «багато-до-багатьох (рис. 1.9).

Однак зв’язок «багато-до-багатьох» безпосередньо не підтримується в реляційних СКБД, наприклад, у СКБД Access. Такий зв’язок потрібно замінити на зв’язки «один-до-багатьох» шляхом використання зв’язкової таблиці (рис. 1.10).

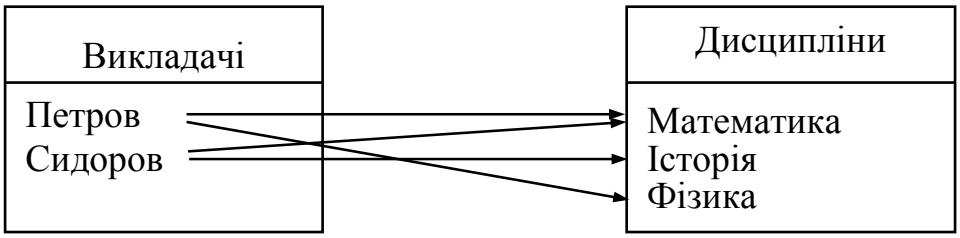


Рис. 1.9 – Організація зв’язку «багато-до-багатьох»

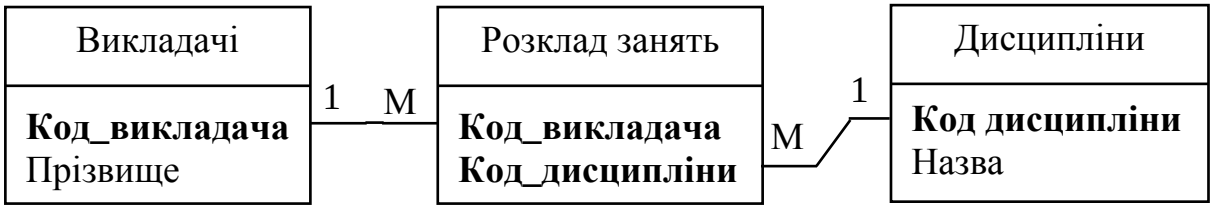


Рис. 1.10 – Спосіб переходу від зв’язку «багато-до-багатьох» до зв’язків «один-до-багатьох»

Індекси та їх призначення

Припустимо, що ми працюємо з кадровою системою. Як знайти інформацію про певного співробітника? У таблиці «Співробітники» записи звичайно йдуть у довільному порядку. Час пошуку потрібних записів простим перебором збільшується. Щоб скоротити час пошуку, необхідно перед початком пошуку впорядкувати записи таблиці за тими полями, які використаються для пошуку (наприклад, за полем «Прізвище»), але при цьому всі записи таблиці треба буде багаторазово переміщати в пам’яті, що потребує додаткових витрат часу.

Для скорочення часу на впорядкування записів доцільно індексувати поля таблиці, які використаються для пошуку (наприклад, поле «Прізвище»). Під час сортування змінюється лише порядок розміщення індексів, а записи таблиці не переміщаються в пам’яті. Час на пошук даних значно скорочується.

Індекси створюються автоматично, якщо при формуванні таблиці задати ознаку «Індексоване поле» для тих полів, за якими передбачається пошук. При наявності цієї ознаки СКБД створює внутрішню таблицю для кожного індексу. Таблиця індексу має два стовпці:

1. значення виразу, що об’єднує всі поля, за якими виконується пошук;
2. місцеположення (адреса) кожного запису в початковій таблиці.

На рис. 1.11 показано принцип сортування таблиці «Викладачі» за індексованим полем «Прізвище» (другий стовпець таблиці, індекс = 2).

При створенні таблиці індексу в стовпці «Покажчик» формуються адреси записів таблиці «Викладачі» (див. таблицю індексу до сортування). Сортування таблиці «Викладачі» за алфавітом поля «Прізвище» виконується автоматично. Адреси записів у полі «Покажчик» розміщуються відповідно до алфавіту поля «Прізвище» (див. таблицю індексу після сортування). Наприклад, «Адреса 3» запису «Абаків» розміщується в першому рядку таблиці індексу, «Адреса 1» запису «Іванов» – у другому рядку, а «Адреса 2» запису «Петров» – у третьому рядку. За цими адресами СКБД здійснює швидкий пошук записів у таблиці «Викладачі».

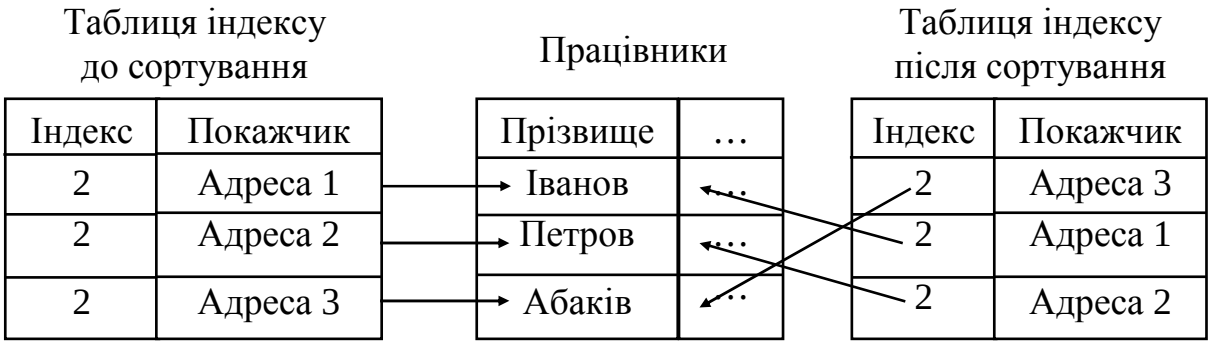


Рис. 1.11 – Принцип сортування за індексованим полем «Прізвище»

Таким чином, завжди доцільно індексувати поля таблиць, за якими передбачається пошук інформації у базі даних. Операція створення індексу називається індексуванням. Ключові поля завжди треба індексувати.

Цілісність даних у базі даних

Ядром будь-якої бази даних є модель даних. Вона становить множину структур даних, обмежень цілісності й операцій маніпулювання даними. За допомогою моделі даних можуть бути представлені об'єкти предметної області та взаємозв'язки між ними. Найбільш поширене трактування реляційної моделі даних належить К. Дейту. Згідно з трактуванням Дейта у склад реляційної моделі входять три частини: структурна, цілісна та маніпуляційна.

Структурна частина реляційної моделі

Структурна частина описує об'єкти, які розглядаються реляційною моделлю, а єдиною структурою даних, яка використовується в реляційних базах даних, є нормалізоване *n*-арне відношення.

Нехай задані n множин даних D_1, D_2,..., D_n, які є доменами. Тоді відношенням R називається множина n впорядкованих кортежів $<d_1, d_2,..., d_n>$, де $d_1 \in D_1$ (символ $\in$ означає «елемент з»), $d_2 \in D_2$,..., $d_n \in D_n$. Кортеж становить підмножину пар {<ім'я атрибуту>, <значення>} і є унікальним у відношенні. У математичному сенсі відношення R є підмножиною декартового добутку доменів $D_1 \times D_2 \times ... \times D_n$. Відношення R включає заголовок і тіло.

Заголовок (схема) відношення містить фіксовану кількість імен атрибутів відношення. Імена атрибутів мають бути унікальними в межах відношення і можуть збігатися з іменами відповідних доменів. Заголовок не змінюється під час роботи з базою даних. Якщо у відношенні змінені, додані або видалені атрибути, то маємо інше відношення (навіть з колишнім ім'ям).

Тілом відношення є набір кортежів. Воно може бути модифікованим під час роботи з базою даних, тобто кортежі можуть змінюватися, додаватися і видалятися.

Кількісні характеристики відношення

Ступінь (арність) відношення – це число атрибутів відношення. Відношення ступеня один називають унітарним, ступеня два – бінарним, ступеня три – тернарним, а ступеня n – n-арним (відношення має n атрибутів). Ступінь відношення не змінюється в часі.

Кардинальність (потужність) відношення – це кількість кортежів відношення. Кардинальне число відношення змінюється в часі. Наприклад, відношення «Викладачі» (рис. 1.6) є 4-арним, а його кардинальність – 4.

Фундаментальні властивості відношень

Властивості відношень безпосередньо виходять з наведеного вище теоретико-множинного визначення поняття відношення, яке є множиною кортежів. Фундаментальними властивостями відношень є:

1.**Однотипність кортежів.** Усі елементи відношення є однотипними кортежами, що дозволяє вважати кортежі аналогом рядків у таблиці.

2.**Обмеженість відношень.** Відношення включає не всі можливі кортежі з декартового добутку доменів, на яких воно визначене. Для кожного відношення є критерій, який дозволяє визначити ті кортежі, що входять у відношення. Цей критерій визначає сенс або семантику відношення, є логічним виразом і називається предикатом відношення R.

3.**Відсутність кортежів-дублікатів і впорядкованості кортежів.** У відношенні відсутні однакові кортежі і кортежі не впорядковані зверху вниз, оскільки відношенням є множина кортежів, а згідно з теорією множин кожна множина складається з різних елементів, які не впорядковані. Звідси витікає, що посилання на кортежі у відношенні повинно мати первинний ключ.

4.**Відсутність впорядкованості атрибутів.** Атрибути відношень не впорядковані, оскільки за визначенням кортеж становить підмножину пар {<ім'я атрибуту>, <значення>} і для посилання на значення використовується ім'я атрибуту. Ця властивість дозволяє модифікувати схеми існуючих відношень шляхом додавання або видалення атрибутів.

5.**Атомарність значень атрибутів**. Значення всіх атрибутів повинні бути атомарними. Це є наслідком визначення домену як потенційної множини значень простого типу, тобто серед значень домену не може бути відношення.

Потенційний, альтернативний і первинний ключі відношення

Потенційний ключ відношення – це мінімальний набір атрибутів, який може бути використаний для однозначної ідентифікації будь-якого кортежу відношення. Ключі відношень бувають атомарними і складеними. Атомарний ключ складається з єдиного атрибуту, а **складений ключ** – із набору атрибутів. Складений ключ не повинен мати зайвих атрибутів. У його склад включаються тільки ті атрибути, без яких не можлива ідентифікація кортежів.

Відношення має принаймні один потенційний ключ. Дійсно, якщо ніякий атрибут або група атрибутів не є потенційним ключем, то через унікальність кортежів усі атрибути разом утворюють потенційний ключ.

Відношення може мати декілька потенційних ключів. Один із них оголошується первинним ключем (ПК), а інші – альтернативними ключами.

Поняття потенційного ключа є семантичним і відображає трактування понять предметної області. Наприклад, розглянемо відношення «ВИКЛАДАЧ», яке подане у вигляді таблиці на рис. 1.6.

На перший погляд може здатися, що в цьому відношенні є три потенційних ключі: «Таб_номер», «Прізвище» і «Зарплата». У кожній колонці таблиці містяться унікальні дані. Проте серед викладачів можуть бути викладачі з однаковою зарплатою і однаковим прізвищем. Табельний же номер по суті, є унікальним для кожного викладача, і саме розуміння семан-

тики даних привело до твердження, що в даному відношенні є тільки один потенційний ключ – «Таб_номер».

Таким чином, потенційні ключі служать єдиним засобом адресації кортежів у відношенні. Тільки знання значень потенційного ключа кортежу дозволяє точно визначити первинний ключ відношення.

У практичній реалізації реляційної СКБД відношення є математичним аналогом таблиць. Таблиця є найбільш звичним і зручним вмістищем для зберігання інформації, має фіксовану кількість пойменованих і впорядкованих стовпців та необмежену кількість рядків. У табл. 1.2 наведені терміни, які вживаються як синоніми.

Таблиця 1.2 – Порівняння термінології

Реляційний термін	**Табличний термін у СКБД**
Відношення	Таблиця
Заголовок відношення	Заголовок таблиці
Тіло відношення	Тіло таблиці
Атрибут відношення	Найменування стовпця (поля) таблиці
Кортеж відношення	Рядок (запис) таблиці
Ступінь відношення	Кількість стовпців таблиці
Кардинальність відношення	Кількість рядків таблиці
Домен	Тип даних (базовий або користувача)

Цілісна частина реляційної моделі

Під цілісністю даних розуміється забезпечення достовірності й узгодження даних, а також виконання логічних обмежень на дані у базі даних у будь-який момент часу її функціонування.

У цілісній частині моделі даних фіксуються дві базові вимоги цілісності, що повинні підтримуватися в будь-якій реляційній СКБД. Перша вимога називається **вимогою цілісності сутностей**, а друга – **вимогою цілісності за посиланнями, або вимогою цілісності зовнішніх ключів**.

Цілісність сутностей. Під сутністю розуміється деякий об'єкт, що становить інтерес для опису предметної області при створенні моделі бази даних. Формою подання сутності є відношення або таблиця.

Правило цілісності сутностей: кожна сутність, що реалізована відношенням, повинна мати первинний ключ, а вхідні до його складу атрибути не можуть приймати NULL-значень. Цей потенційний ключ краще всього оголошувати первинним ключем таблиці даної суті.

Примітка. NULL-значення – це, власне, не значення, а маркер, який показує, що значення невідоме. Проблема використання NULL-значення в теорії реляційних баз даних остаточно не вирішена. Практично всі реалізації сучасних реляційних СКБД дозволяють використовувати NULL-значення, незважаючи на їх недостатню теоретичну обґрунтованість. Безумовно, слід уникати появи NULL-значень, обумовлених некоректним проектуванням бази даних, але в реальній ситуації, коли якісь дані про екземпляр сутності невідомі, без NULL-значень не обійтися.

Цілісність зовнішніх ключів. Цілісність зовнішніх ключів означає підтримку зв’язків між записами пов'язаних таблиць при внесенні змін у базу даних. Цілісність даних забезпечується при виконанні таких умов:

1. Пов'язане поле головної таблиці повинно бути ключовим полем.
2. Пов'язані ключові поля головної та підлеглої таблиць повинні мати однаковий тип даних, а імена цих полів можуть бути різними.
3. Пов'язані таблиці повинні бути таблицями однієї бази даних.
4. Для забезпечення цілісності зовнішнього ключа необхідно в процесі створення та експлуатації бази даних враховувати такі правила:

- не можна вводити в поле зовнішнього ключа підлеглої таблиці значення, яке відсутнє в ключовому полі головної таблиці. Наприклад, не припустимо вводити в ключове поле «Код_викладача» таблиці «Дисципліни» значення, якого немає в таблиці «Викладачі» (див. рис. 1.8);
- не допускається видалення запису з головної таблиці, якщо існують пов'язані з нею записи в підлеглій таблиці. Наприклад, неможливо видалити запис із таблиці «Викладачі», якщо в таблиці «Фотографії» існують записи, які пов'язані з даним співробітником;
- не можна змінювати значення ключового поля в головній таблиці, якщо існують записи, пов'язані з даним значенням. Наприклад, не можна змінювати код співробітника в таблиці «Викладачі», якщо в таблиці «Фотографії» є записи, що відносяться до цього співробітника.

У СКБД для реалізації цих правил передбачений механізм забезпечення цілісності даних. При порушенні умов цілісності дія не виконується й на екран виводиться попередження. Крім того, для реалізації обмежень на зміну або видалення пов'язаних записів у СКБД передбачено механізм каскадного оновлення або каскадного видалення. Цілісність даних зберігається завдяки тому, що СКБД при зміні ключового поля головної таблиці автоматично змінює й відповідні значення пов'язаних записів, а при видаленні за-

пису в головній таблиці – видаляє й усі пов'язані записи в підлеглій таблиці.

Маніпуляційна частина реляційної моделі

Важливою частиною реляційної моделі даних є механізм маніпулювання даними, який забезпечує вибірку й оновлення даних. У маніпуляційній частини моделі стверджується два фундаментальних механізми маніпулювання реляційними базами даних: реляційна алгебра і реляційне числення. Перший механізм базується на класичній теорії множин, а другий – на класичному логічному апараті числення предикатів першого порядку.

Неформально реляційну алгебру можна визначити як високорівневу процедурну мову – мову запитів. За допомогою запитів користувач повідомляє СКБД правила побудови необхідного йому відношення з одного або декількох відношень, що існують у базі. Реляційна алгебра має велику виразну потужність: дуже складні запити до бази даних можуть бути виражені за допомогою одного виразу реляційної алгебри.

Реляційне числення не є процедурною мовою, яку можна використовувати для визначення відношення, що може буде створене на основі одного або декількох інших відношень бази даних. І хоча в реляційній моделі даних присутні обидва механізми, далі розглянемо тільки операції реляційної алгебри.

Склад операцій і замкнутість реляційної алгебри

Реляційна алгебра – це мова операцій, що дозволяють створювати на основі одного або декількох відношень інше відношення без зміни початкових відношень.

Замкнутість реляційних операцій

Реляційні операції є *замкнутими* щодо відношення, оскільки операндами та результатами операцій реляційної алгебри є відношення. Ця властивість реляційних операцій дозволяє створювати вкладені вирази реляційної алгебри (за аналогією з тим, як створюються вкладені арифметичні вирази). При будь-якій глибині вкладеності виразів результатом обчислень завжди є відношення.

Склад операцій реляційної алгебри

Теоретичну основу реляційної алгебри складає теорія множин, оскільки відношення також є множинами. У склад реляційної алгебри, яку запропонував Кодд, включено вісім основних операцій. Вони діляться на два класи – теоретико-множинні та спеціальні реляційні операції.

Теоретико-множинними є такі операції: об'єднання відношень, перетин відношень, різниця відношень і декартовий добуток відношень. Спеціальними реляційними операціями є такі операції: вибір (селекція) відношень, проекція відношень, з'єднання відношень, ділення відношень. Операції вибору і проекції є **унарними**, а інші – **бінарними**. Унарна операція здійснюється над одним відношенням, а бінарна – над парою відношень.

Крім основних операцій, у склад реляційної алгебри включено операції присвоювання та перейменування.

Операція присвоювання (=) призначена для зберігання результату обчислення реляційного виразу в існуючому відношенні бази даних.

Операція перейменування (RENAME) призначена для зміни імен атрибутів. Результатом цієї операції є нове відношення з новими іменами атрибутів. Синтаксис операції перейменування:

$$R \text{ RENAME } A_1, A_2,\ldots, A_k, NewA_1, NewA_2, \ldots, NewA_k,$$

де R – відношення;

$A_1, A_2,\ldots, A_k$ – початкові імена атрибутів;

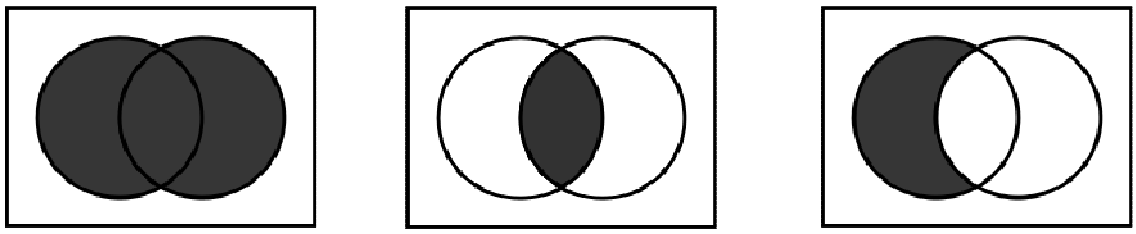
$NewA_1, NewA_2, \ldots, NewA_k$ – нові імена атрибутів.

Теоретико-множинні операції реляційної алгебри

Сенс операцій об'єднання, перетину та різниці в реляційній алгебрі в цілому залишається теоретико-множинним (рис. 1.12):

- об'єднанням двох множин A{a} і B{b} є така множина C{c}, що для кожного c або існує такий елемент a множини A, що c = a, або існує такий елемент b множини B, що c = b (рис. 1.12 а);
- перетином множин A{a} і B{b} є така множина C{c}, що для будь-якого c існують такі елементи a і b, що c = a = b (рис. 1.12 б);
- різницею множин A{a} і B{b} є така множина C{c}, що для будь-якого c існує такий елемент a, що c = a, і не існує такий елемент b, що c = b (рис. 1.12 в).

Для операцій об'єднання, перетину та різниці потрібна сумісність відношень за об'єднанням. Два відношення сумісні за об'єднанням у тому і лише в тому випадку, коли мають однакові заголовки й однойменні атрибути визначені на одному й тому ж домені. Іншими словами, відношення є сумісними за об'єднанням, якщо мають однакову арність, однакові імена атрибутів і однойменні атрибути мають однаковий тип.



а) об’єднання б) пересічення в) різниця

Рис. 1.12 – Ілюстрація результатів теоретико-множинних операцій

Друга особливість реляційних операцій полягає в можливості перейменування атрибутів. Якщо два відношення сумісні у всьому, окрім імен атрибутів, то перед виконанням цих операцій необхідно зробити перейменування атрибутів із використанням операції перейменування.

Операції над відношеннями реляційної алгебри

Теоретико-множинні операції реляційної алгебри

Розглянемо правила виконання теоретико-множинних операцій.

Операція об'єднання (UNION). Позначення – $R \cup S$

Об'єднанням відношень *R* і *S* називається множина кортежів, які належать або *R*, або *S*, або їм обом (рис. 1.13).

R

A	B
a_1	b_1
a_2	b_2

S

A	B
a_1	b_1
a_3	b_3

$R \cup S$

A	B
a_1	b_1
a_2	b_2
a_3	b_3

Рис. 1.13 – Об'єднання відношень $R \cup S$

За допомогою операції об'єднання може бути реалізовано додавання кортежу з наявного відношення. У цьому випадку *R* – початкове відношення, а *S* – відношення, що містить один кортеж, який додається.

Операція пересічення (INTERSECTION). Позначення – $R \cap S$

Пересіченням відношень *R* і *S* називається множина кортежів, що належать як *R*, так і *S* (рис. 1.14).

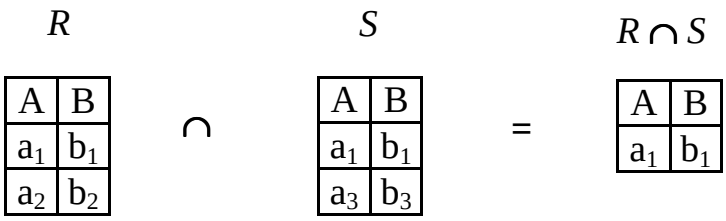


Рис. 1.14 – Пересічення відношень *R* і *S*

Операція різниця (DIFFERENCE). Позначення – R – S

Різницею відношень *R* і *S* називається множина кортежів, які належать *R* і не належать *S* (рис. 1.15). Відношення повинні мати однакову арність.

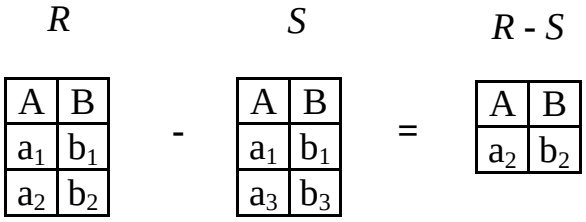


Рис. 1.15 – Різниця відношень *R – S*

Операція декартового добутку (CROSS-PRODUCT) – R × S

Операція декартового добутку застосовується для множення двох відношень. Множенням двох відношень називається створення іншого відношення, що складається зі всіх можливих пар кортежів обох відношень.

Формально операція розширеного декартового добутку визначається так. Нехай є два відношення $R_1(a_1, a_2,..,a_m)$ і $R_2(b_1, b_2,\ldots,b_n)$. Результатом цієї операції є відношення $R(a_1, a_2,\ldots,a_n, b_1, b_2,\ldots,b_m)$, у якому кожний кортеж є об'єднанням одного кортежу першого відношення і одного кортежу другого відношення: $(ra_1, ra_2,\ldots,ra_n, rb_1, rb_2,\ldots, rb_m)$, де $(ra_1, ra_2,\ldots, ra_n)$ – кортеж відношення R_1, а $(rb_1, rb_2,\ldots,rb_m)$ – кортеж відношення R_2. Якщо одне відношення (R_1) має m атрибутів і k кортежів, а інше (R_2) – n атрибутів і p кортежів, то їх декартовий добуток міститиме $(m + n)$ атрибутів і $(k \times p)$ кортежів.

У цій операції відношення повинні бути сумісними. Два відношення **сумісні за взяттям розширеного декартового добутку**, якщо пересічення множин імен атрибутів, узятих із їх схем, буде пустим. Це означає, що імена атрибутів початкових відношень повинні бути унікальними (рис. 1.16).

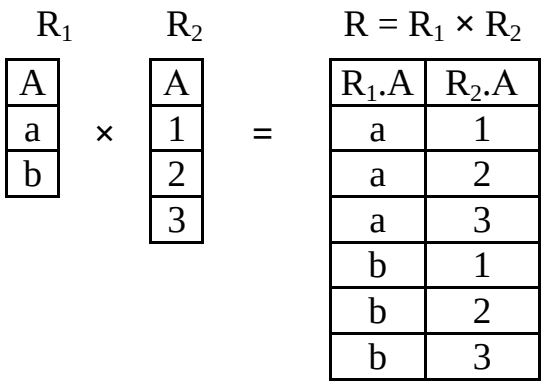


Рис. 1.16 – Декартовий добуток: $R \times S$

Для забезпечення цієї вимоги імена атрибутів у відношенні-результаті R доповнюються назвами початкових відношень у вигляді префіксів.

Слід зауважити, що операція взяття декартового добутку не є дуже осмисленою на практиці, оскільки потужність тіла її результату надто велика.

Основний сенс включення операції декартового добутку до складу реляційної алгебри Кодда полягає в тому, що на її основі визначається дійсно корисна операція з'єднання.

Спеціальні операції реляційної алгебри

До складу спеціальних операцій входять: вибір відношень, проекція відношень, з'єднання відношень та ділення відношень.

Операція вибору (SELECTION). Позначення – σF(R)

Нехай R – відношення, а F – предикат (формула), що утворений операндами, які є константами, іменами атрибутів, арифметичними операторами порівняння або логічними операторами (AND – І, OR – АБО, NOT – НІ). Тоді вибором (селекцією) **σF** називається множина кортежів, компоненти якої задовольняють умову, що задана предикатом F. Ця операція будує нове відношення, яке є «горизонтальною» підмножиною початкового відношення. На рис. 1.17 наведено приклад виконання операції $\sigma_{A = a3}(R)$ вибору кортежів відношення R. Тут F: $A = a_3$ – вміст стовпця A дорівнює значенню a_3. Згідно з умовою вибирається третій кортеж. Отримано нове відношення S, у склад якого входить лише один кортеж.

R

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3	**b_2**	**c_3**	**d_2**
a_4	b_4	c_4	d_4

$\sigma_{A = a3}(R) =$

S

A	B	C	D
a_3	b_2	c_3	d_2

Рис. 1.17 – Виконання операції вибору

Операція проекції (PROJECT). Позначення – $\boldsymbol{\pi_{A_1, A_2, ..., A_m}(R)}$

Проекція $\boldsymbol{\pi_{A_1, A_2, ..., A_m}(R)}$ **–** це множина кортежів, що складається з кортежів відношення *R* шляхом вибору стовпців з іменами A_1, A_2,…, A_m. Ця операція будує «вертикальну» підмножину (*S*) шляхом вибору тільки заданих атрибутів і виключення останніх. Кортежі, що повторюються, також видаляються. Результат операції проекції $\pi_{B, D}(R)$ показано на рис. 1.18.

R

A	**B**	C	**D**
a_1	**b_1**	c_1	**d_1**
a_2	**b_2**	c_2	**d_2**
a_3	**b_2**	c_3	**d_2**
a_4	**b_4**	c_4	**d_4**

$\pi_{B,D}(R) =$

S

B	D
b_1	d_1
b_2	d_2
b_4	d_4

Рис. 1.18 – Проекція: $\mathbf{S = \pi_{B, D}(R)}\, S$

Операції з'єднання (JOIN). Позначення – R ⊳⊲ S

Як правило, користувачів цікавить лише деяка частина всіх комбінацій кортежів декартового добутку, яка задовольняє задану умову. Тому замість декартового добутку зазвичай використовується одна з найважливіших операцій реляційної алгебри – операція *з'єднання*. У результаті її виконання на базі двох початкових відношень створюється нове відношення.

На практиці застосовуються три різновиди операції з'єднання:

- умовне з'єднання відношень (тета-з’єднання);
- еквіз'єднання відношень (з'єднання за еквівалентністю);
- природне з'єднання відношень.

Умовне (тета-) з'єднання (CONDITIONAL). Позначення – R ⊳⊲ a θ b S

Операція умовного з'єднання здійснюється над двома відношеннями *R* і *S*. Результатом операції є відношення, яке містить кортежі з декартово-

го добутку відношень $R \times S$, що задовольняють предикат F. Предикат F має вигляд $a\ \theta\ b$, де a і b – імена атрибутів різних відношень-операндів, θ – арифметичний оператор порівняння ($=, \neq, >, \geq, <, \leq$).

Формальне визначення операції. З'єднання відношень R і S за стовпцями A_i та A_j визначає в декартовому добутку $R \times S$ множину кортежів, у якій i-й компонент R знаходиться у відношенні θ з j-м компонентом S. Операція умовного з'єднання здійснюється за формулою:

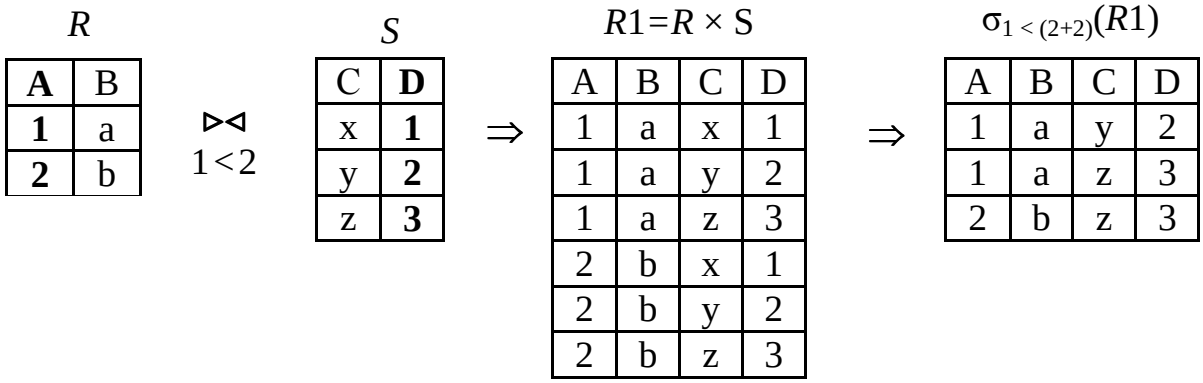
$$\boldsymbol{R} \rhd\lhd_{\boldsymbol{A}}\ \theta\ \mathrm{B}\ \boldsymbol{S} = \sigma_{\boldsymbol{i}}\ \theta\ (_{\boldsymbol{l}}+_{\boldsymbol{j}})(\boldsymbol{R} \times \boldsymbol{S}),$$

де l – арність відношення R.

При реалізації операції з'єднання послідовно виконуються дві операції:

1) операція декартового добутку $R \times S$;
2) операція вибір $\sigma i\ \theta\ (l+j)(R \times S)$.

Нехай необхідно виконати з'єднання двох відношень R і S при умові, що значення першого стовпця ($i = 1$) відношення R повинні бути менше значень другого стовпця ($j = 2$) відношення S (рис. 1.19 а). Для наочності стовпці, значення яких порівнюються, виділені напівжирним написанням.



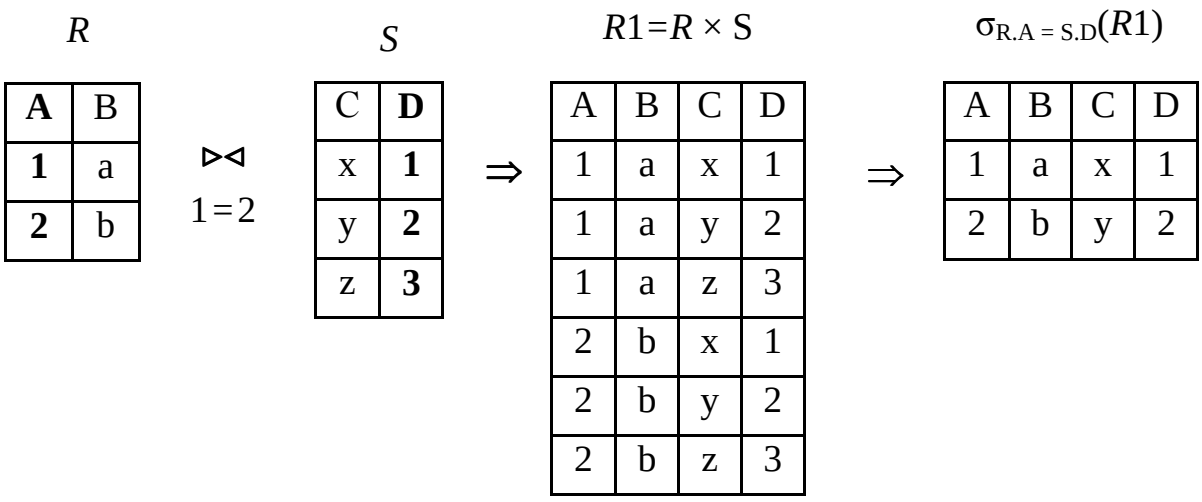
а) б) в)

Рис. 1.19 – Приклад виконання операції з'єднання

Першою виконується операція декартового добутку (рис. 1.19 б), а другою – операція вибору за умовою 1 < (2+2), яка отримана шляхом підстановки в загальну умову $i\ \theta\ (l+j)$ таких значень: $i = 1$, $l = 2$, $j = 2$, θ – <.

Еквіз'єднання (з'єднання за еквівалентністю). **Позначення –** $R \rhd\lhd S_{a=b}$

Операція називається еквіз'єднанням, якщо предикат F містить тільки операцію порівняння за рівністю (=). Приклад виконання операції еквіз'єднання $\sigma_{\mathrm{R.A = S.D}}(R \times S)$ наведено на рис. 1.20.



а) б) в)

Рис. 1.20 – Приклад виконання операції еквіз'єднання

Операція еквіз'єднання найчастіше зустрічається на практиці і для неї існують найбільш ефективні алгоритми реалізації.

Природне з'єднання. **Позначення – *R ⋈ S***

Ця операція застосовна до пари відношень, у яких збігаються імена деяких атрибутів, наприклад, імена зовнішніх ключів (ключів зв'язку). **Природним з'єднанням** називається з'єднання за еквівалентністю двох відношень, що виконане за всіма загальними атрибутами, з результатів якого виключається по одному екземпляру кожного загального атрибуту.

Нехай задані два відношення R і S:

$$R = (A_1 \ldots, A_k, B_1 \ldots, B_n),\ S = (A_1, \ldots, A_k, C_1, \ldots, C_m),$$

де імена $A_1, \ldots, A_k$ збігаються.

Тоді операція природного з'єднання $R \bowtie S$ визначається так:

$$R \bowtie S = \pi_{A1,\ldots, Ak, B1,\ldots,Bn, C1,\ldots, Cm}(\sigma_{R.A1=S.A1\&R.A1=R.A2\&S.A2,\ldots, R.Ak = S.Ak}(R \times S)).$$

В операції природного з'єднання послідовно виконуються такі операції:

1. операція декартового добутку: $R1 = R \times S$ з перейменуванням атрибутів, імена яких збігаються;
2. операція еквіз'єднання: $R2=\sigma_{R.A1=S.A1\ \&\ R.A1=R.A2\ \&\ S.A2,\ldots,\ R.Ak = S.Ak}(R1)$;
3. операція проекції $R3 = \pi_{A1,\ldots, Ak, B1,\ldots,Bn, C1,\ldots, Cm}(R2)$.

Приклад виконання операції природного з'єднання $R \bowtie S$ наведено на рис. 1.21.

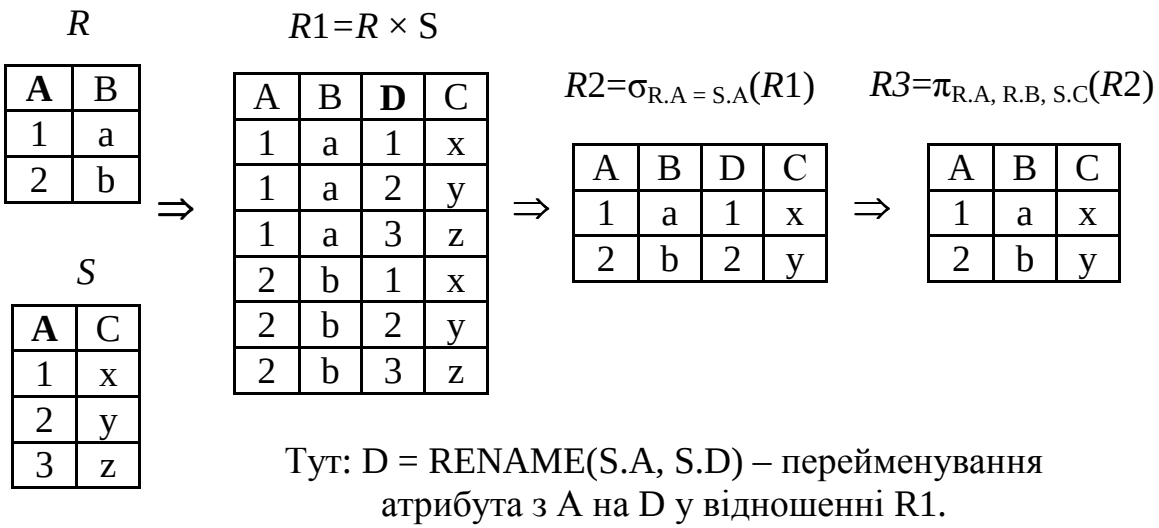


Рис. 1.21 – Приклад виконання операції природного з'єднання

Приклад операції з'єднання

Нехай у базі даних «Екзамени» зберігаються результати екзаменів студентів групи 2010 з математики й інформатики (рис. 1.22).

Рис. 1.22 – Таблиці бази даних «Екзамени»

Необхідно визначити прізвища студентів, які склали всі екзамени на відмінно.

Для розв'язання задачі виконаємо таку послідовність операцій.

1. Операція вибору $R1 = \sigma_{\text{Оцінка} = 5}$ (Екзамени). Отримаємо відношення *R*1, у якому будуть тільки оцінки відмінно (рис. 1.23).

$R1 = \sigma_{\text{Оцінка} = 5}(\text{Екзамени})$

КодСт	**КодДисц**	Оцінка
1001	101	5
1001	102	5
1003	101	5
1003	102	5

Рис. 1.23 – Результат операції вибору

2. Операція еквіз'єднання відношень *Студенти* і *R*1 (рис. 1.24)

$R2 = \sigma_{\text{Студенти.КодСт} = \text{Оцінка.Код}}\text{Ст}$ (*Студенти* × *R*1).

$R2 = \sigma_{\text{Студенти.КодСт} = \text{Оцінка.КодСт}}$ (*Студенти* × *R*1)

КодСт	Прізвище	**КодСт1**	**КодДисц**	Оцінка
1001	Іванов	1001	101	5
1001	Іванов	1001	102	5
1002	Петров	1003	101	5
1002	Петров	1003	102	5

Рис. 1.24 – Результат операції еквіз'єднання відношень *Студенти* і *R*1

3. Операція проекції відношення *R*2 (рис. 1.25)

$R3=\pi_{R2.\text{КодСт}, R2.\text{Прізвище}, R2.\text{КодДисц}, R2.\text{Оцінка}}(R2)$.

$R3=\pi_{R2.\text{КодСт}, R2.\text{Прізвище}, R2.\text{КодДисц}, R2.\text{Оцінка}}(R2)$

КодСт	Прізвище	**КодДисц**	Оцінка
1001	Іванов	101	5
1001	Іванов	102	5
1002	Петров	101	5
1002	Петров	102	5

Рис. 1.25 – Відношення *R*3 після операції проекції відношення *R*2

4. Операція природного з'єднання відношень *R*3 і *Дисципліни* (рис. 1.26):

$$R5=\pi_{R3.\text{КодСт}, R3.\text{Прізвище}, R3.\text{Оцінка}, \text{Дисципліни.Назва}}(R4),$$

де $R4 = (\sigma_{R3.\text{КодДисц} = \text{Дисципліни.КодДисц}}(R3 \times \textit{Дисципліни})$.

Отримаємо відношення *R*5, у якому будуть дані про студентів,

які отримали оцінки за всіма дисциплінами (рис. 1.26).

$R5=\pi_{R3.КодСт,\ R3.Прізвище,\ R3.Оцінка,\ Дисципліни.НазваДисц}(R4)$

КодСт	Прізвище	Оцінка	НазваДисц
1001	Іванов	5	Математика
1001	Іванов	5	Інформатика
1003	Петров	5	Математика
1003	Петров	5	Інформатика

Рис. 1.26 – Студенти, які отримали оцінки «відмінно»

Можливі й інші варіанти розв’язання задачі. Спробуйте самостійно знайти інше вирішення поставленої задачі.

Операція з'єднання має велике значення для реляційних баз даних, оскільки при виконанні запитів користувача вона дозволяє з’єднати декілька відношень в одне відношення.

Ділення відношень R ÷ S

Ця операція найменш очевидна з усіх операцій реляційної алгебри Кодда і потребує докладнішого пояснення. Операція виконується над двома відношеннями $R(a_1, a_2,...,a_n, b_1, b_2,...,b_m)$ і $S(b_1, b_2,...,b_m)$, що мають у загальному випадку різні структури та частину однакових атрибутів $b_1, b_2,...,b_m$. Вважатимемо, що атрибути b_i відношень R і S (i = 1, 2,…,m) мають не тільки однакові імена, але й визначені на одному й тому ж домені, наприклад, це ключові атрибути відношень.

Назвемо множину атрибутів $\{a_j\}$ складеним атрибутом A, а множину атрибутів $\{b_j\}$ – складеним атрибутом B. Після цього розглянемо реляційне ділення «бінарного» відношення $R(A, B)$ на унарне відношення $S(B)$. Результатом ділення відношення $R(A, B)$ на відношення $S(B)$ є «унарне» відношення $C(A)$. Таким чином, у результаті операції ділення утворюється нове відношення C шляхом виключення з множини атрибутів відношення R множини атрибутів відношення S.

Кортежі відношення C складаються з множини кортежей відношення R, які визначені на атрибуті A та відповідають комбінації всіх кортежей відношенням S (рис. 1.27).

Розглянуті вище операції реляційної алгебри реалізуються в мові маніпулювання даними СКБД, що забезпечує обробку реляційних таблиць. До таких мов відносяться, наприклад, мова SQL (Structured Query

Language) і мова QBE (Query By Example).

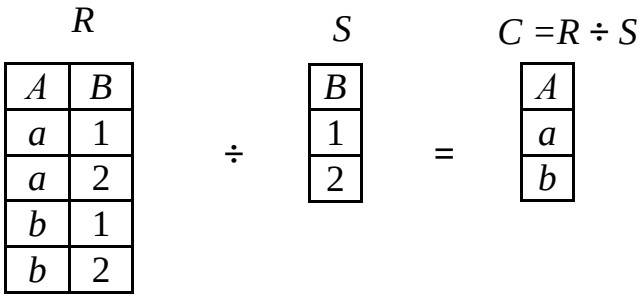


Рис. 1.27 – Операція ділення відношень *R* на *S*

Мова SQL, що реалізована в більшості СКБД, є реляційно повною, оскільки, крім операцій реляційної алгебри, вона містить повний набір операторів над кортежами – *Включити, Видалити, Змінити*, а також реалізує арифметичні операції та операції порівняння.

Семантика мови SQL багато в чому базується саме на реляційній алгебрі, що спрощує ознайомлення та вивчення цієї мови.

Питання для самодіагностики

1. Дайте характеристику структурної, цілісної та маніпуляційної частин реляційної моделі даних.
2. Дайте загальну характеристику операцій реляційної алгебри.
3. Дайте характеристику різновидам операції з'єднання.
4. Чим відрізняються операції еквіз'єднання та природного з'єднання?
5. Дайте характеристику операції умовного з'єднання. Наведіть приклад виконання операції умовного з'єднання.
6. Дайте характеристику операції проекції. Наведіть приклад виконання операції проекції.
7. Яку послідовність операцій необхідно виконати, щоб визначити прізвища студентів, які склали всі екзамени на добре та відмінно (рис. 1.22)?

1.3 Інфологічна модель даних

Поняття про інфологічну модель даних

Процес проектування бази даних тривалий, вимагає обговорень із замовником і фахівцями в предметній області. При розробці корпоратив-

них інформаційних систем проект бази даних є тим фундаментом, на якому будується вся система в цілому. Питання про можливість кредитування часто вирішується експертами банку на підставі грамотно зробленого інфологічного проекту бази даних. Отже, інфологічна модель повинна включати такий формалізований опис предметної області, який буде легко «читатися» як фахівцями з баз даних, так і кінцевими користувачами, а також буде придатним для оцінки глибини і коректності опрацьовування проекту бази даних без прив'язки до конкретної СКБД.

Проблема представлення семантики даних є дуже важливою і актуальною для розробників інформаційних систем. Головним призначенням інфологічного моделювання є забезпечення можливості вираження семантики даних. Потреби проектувальників баз даних в зручних засобах моделювання предметної області сприяли виникненню напряму інфологічного, або семантичного, моделювання даних. У 70-х роках XX сторіччя було запропоновано декілька моделей даних, які були названі інформаційними або семантичними, моделями.

Однією з найбільш популярних інформаційних моделей даних є модель «сутність–зв’язок», або ER-модель (Entity Relationship). На використанні різновидів ER-моделі заснована більшість сучасних підходів до проектування реляційних баз даних. Модель була запропонована Ченом у 1976 році. У даний час модель Чена «сутність – зв’язок» стала фактичним стандартом при інфологічному моделюванні баз даних. Загальноприйнятою стала скорочена назва ER-модель.

Моделювання предметної області базується на використанні графічних діаграм – ER-діаграм (Entity Relationship Diagram), що включають невелике число різнорідних компонентів. На основі аналізу предметної області визначаються основні об'єкти предметної області, будується схема бази даних у вигляді ER-діаграм. На схемі в графічній формі відображаються зв'язки між об’єктами та характеристики цих зв'язків. Потім за чіткими правилами здійснюється перехід від ER-діаграм до таблиць бази даних, здійснюється наповнення таблиць атрибутами і перевірка їх на виконання умов нормалізації. Визначаються ключові атрибути таблиць та зв'язки між таблицями. Результатом проектування є схема реляційної бази даних.

У зв'язку з наочністю подання концептуальних схем баз даних ER-моделі набули широкого поширення в CASE-системах, що підтримують автоматизоване проектування реляційних баз даних. Більшість сучасних

CASE-систем містять інструментальні засоби для опису даних у формалізмі моделі «сутність – зв’язок» та реалізують методи автоматичного перетворення проекту бази даних із ER-моделі в реляційну модель. Перетворення виконується в логічну модель, яка відповідає конкретній СКБД. Прикладом CASE-системи є проста й універсальна програма ERwin фірми PLATINUM, яка призначена для автоматизованого створення реляційних баз даних.

Усі CASE-системи мають розвинені засоби документування процесу розробки бази даних. Автоматичні генератори звітів дозволяють підготувати звіт про поточний стан проекту бази даних з докладним описом об'єктів бази даних і їх відношень як у графічному вигляді, так і у вигляді готових стандартних друкарських звітів, що істотно полегшує ведення проекту.

Зараз не існує єдиної загальноприйнятої системи позначень для ER-моделі, і різні CASE-системи використовують різні графічні нотації, але знання однієї з них, наприклад нотації Чена, дозволяють легко зрозуміти й інші нотації.

Мета інфологічного проектування предметної області

Метою інфологічного моделювання є забезпечення розробника бази даних концептуальною схемою бази даних на рівні уявлень про предметну область.

Як інструмент інфологічного моделювання використовуються різні варіанти ER-діаграм. З їх допомогою визначаються важливі для предметної області об'єкти (сутності), їх властивості (атрибути) і стосунки один з одним (зв'язки). Інфологічне моделювання відображає той факт, що сутності мають зв’язки між собою, а атрибути належать сутностям.

Інфологічна модель даних відображає дані з погляду їх сенсу і в контексті користувача. Модель приховує технічні деталі й підкреслює найбільш важливу суть з погляду предметної області і користувача. Спосіб зображення моделі – діаграми «сутність – зв'язок» (ER-діаграми) будь-якого з наявних типів.

Найчастіше на практиці інфологічне моделювання використовується на стадії інфологічного проектування бази даних. При цьому в термінах моделі проводиться розробка концептуальної схеми бази даних, яка потім уручну перетворюється на реляційну схему. Цей процес виконується з використанням методик, у яких досить чітко обумовлені всі етапи такого перетворення.

Основні терміни та визначення щодо поняття сутності

Модель «сутність–зв’язок» має декілька базових понять, які утворюють початкові цеглинки, з яких будуються вже складніші об'єкти за заздалегідь визначеними правилами. Основними конструктивними елементами моделі «сутність-зв’язок» є: сутність, екземпляр сутності, атрибути, зв’язки та асоціативні сутності.

Алгоритм визначення сутностей

Сутність – це клас однотипних об'єктів, інформація про яких має бути врахована в моделі бази даних. Оскільки сутність відповідає деякому класу однотипних об'єктів, то передбачається, що в системі існує безліч екземплярів даної сутності. Наприклад, як сутність можуть виступати люди, об'єкти, події та ролі, які всі вони грають. Ці об'єкти повинні мати екземпляри, що відрізняються один від одного і допускають однозначну ідентифікацію.

Екземпляр сутності – це конкретний представник даної сутності. Кожна сутність повинна мати унікальне в рамках даної моделі ім’я, яке рекомендується задавати іменником в однині. Наприклад, сутність ПРЕДМЕТ («Предмет для вивчення») має екземпляр – «Математика» (конкретний предмет), а для сутності ГРУПА – екземпляром є, наприклад, ГІС 2007.

Об'єкт, якому відповідає поняття сутності, має свій набір атрибутів – характеристик, що визначають властивості даного представника класу.

Атрибут сутності – це іменована характеристика деякої властивості сутності або зв’язку. Найменування атрибуту має бути виражене іменникам в однині (можливо, з прикметниками, що характеризують атрибут). Атрибут залежить тільки від сутності та ні від чого іншого. Він може мати тільки одне значення в даний момент часу. Наприклад, у сутності ПРАЦІВНИК може бути такий набір атрибутів: Табельний номер, Прізвище, Ім'я, По батькові, Дата народження, Кількість дітей.

При проектуванні атрибутів корисно ставити такі запитання:

- Які дані про сутність ми хочемо зберігати?
- Які властивості є в екземплярі цієї сутності?
- Чи є в екземплярі сутності тільки один екземпляр цієї сутності?
- Чи може змінюватися описана атрибутом характеристика сутності з часом?

Наприклад, у людини завжди є тільки одна стать, у будівлі – тільки одна висота в даний момент часу, у людини завжди є зростання віку.

Імена атрибутів рекомендується робити *унікальними* в рамках всієї схеми бази даних. Оскільки атрибут описує тільки одну сутність, він і має бути у всій схемі один (за правилом «один факт – в одному місці»).

Розглянемо деякі правила, за якими корисно визначати атрибути.

- *Атрибут повинен описувати саме свою сутність*. Це просте правило не так просто застосовувати на практиці. Утруднення виникають, коли в наявній частині моделі виявляється новий атрибут, і треба знайти для нього відповідну сутність. Якщо сутність відсутня, то модель доведеться змінити.
- *Не варто об'єднувати в один атрибут різнорідні дані*. Це найчастіше відноситься до спроб з'єднати в одному атрибуті кількість і одиницю вимірювання, наприклад, зберігати в одному атрибуті щось подібне до «20 ящиків» і «100 пляшок» або «100 грн» і «10 доларів». Слід розділяти подібні атрибути на атрибути для зберігання кількості та атрибути для зберігання одиниці вимірювання.
- *Не рекомендується створювати атрибути, які можуть мати різний сенс залежно від свого конкретного значення*. Наприклад, атрибут «клієнт» може містити дані про фізичну або юридичну особу. Зрозуміло, що ці дані виявляться абсолютно різними за своїм сенсом, і без інформації про те, до якого типу відноситься клієнт, трактувати їх надто складно.
- *Не варто задавати «особливих» діапазонів значень і правил трактування даних.* Тобто слід уникати ситуацій, коли, наприклад, атрибут оцінка містить оцінку, отриману студентом на іспиті (число 3, 4 або 5), у тому випадку, якщо іспит складений успішно; NULL-значення, якщо термін складання іспиту ще не наступив, і негативне число, яке означає, що іспит здавався кілька разів (модуль значення), але так і не був зданий.
- *Не варто задавати атрибути*, для яких важливо зберігати додаткову інформацію. Тут потрібно застосовувати просте правило: *якщо необхідно зберігати додаткову інформацію про атрибут, значить це і є сутність.* Така сутність зазвичай називається *атрибутивною сутністю.* Приклад атрибутивної сутності: аудиторія, в якій відбуваються заняття, могла б бути атрибутом сутності «Проведення заняття», але якщо потрібно зберігати характеристики аудиторії, такі, як місткість, наявність дошки, то доцільно зробити її атрибутивною сутністю.

Зв'язки

Зв'язок – це з'єднання між двома або більше сутностями. Зазвичай зв'язок виражається дієсловом. *Екземпляром зв'язк*у є конкретний зв’язок між конкретними екземплярами сутностей.

Приклади опису зв'язків між сутностями (екземплярами сутностей):

- студент *входить* до групи (Іванов *належить* до групи ГІС2007);
- працівник *може мати* декілька дітей (Петров *має* двох дітей).

Зв'язок між двома сутностями називається *бінарним*. Це основний тип зв'язку, за допомогою якого виражаються зв'язки іншого типу.

Для кожного з двох кінців бінарного зв'язку задаються основні характеристики зв'язку:

- *назва*, яка описує семантику зв'язку, з погляду однієї із сутності;
- *кардинальне число (ступінь, потужність)*, яке описує кількість можливих зв'язків для кожної із двох сутностей;
- *ступінь участі сутності у зв'язку (обов'язковість зв'язку, модальність зв'язку, клас належності)*.

Бінарний зв'язок буває «один-до-одного» (1:1), «один-до-багатьох» (1:N) або «багато-до-багатьох» (N:M). Тип зв’язку задається опцією, яка називається потужністю, або кардинальністю, зв'язку. Крім того, зв'язок характеризується таким параметром, як модальність, який може набувати всього два значення: обов'язковий чи необов'язковий зв’язок.

Визначення первинного та вторинного ключів

Алгоритм визначення ключів відношення

Набір атрибутів сутностей має бути таким, щоб можна було розрізняти конкретні екземпляри сутності.

Ключем сутності є ненадмірний набір атрибутів, значення яких у сукупності є *унікальними* для кожного екземпляра сутності. Ненадмірність полягає в тому, що видалення будь-якого атрибуту з ключа порушує його унікальність.

Ключ (можливий ключ, потенційний ключ) сутності – це мінімальний набір атрибутів, який може бути використаний для однозначної ідентифікації будь-якого екземпляру сутності. Ключі сутностей бувають атомарними і складеними. Атомарний ключ складається з єдиного атрибуту. Складеним ключем є ненадмірний набір атрибутів (ключ не повинен містити зайвих атрибутів). Це означає, що якщо будь-який атрибут виключити

із ключового набору, то атрибутів, що залишилися, буде вже недостатньо для ідентифікації кортежу.

Будь-яка сутність має принаймні один потенційний ключ. Дійсно, якщо ніякий атрибут або група атрибутів не є потенційним ключем, то через унікальність екземплярів сутності всі атрибути разом утворюють потенційний ключ. Набори ключових атрибутів сутності визначаються на ранніх стадіях проектування бази даних.

Наявність *ключа сутності* забезпечує виконання основного правила для сутності: *кожен екземпляр сутності повинен бути унікальним і чітко відрізнятися від будь-якого іншого екземпляра цієї сутності і від будь-яких екземплярів інших сутностей*. Тим самим виконується правило, що *кожне явище або об'єкт може бути представлено тільки однією сутністю*. Не може бути двох однакових сутностей.

Під алгоритмом визначення ключів будемо розуміти правила, за якими здійснюється вибір ключів відношення.

Первинним ключем сутності може бути призначений один із її потенційних ключів. При цьому слід брати до уваги те, що на фізичному рівні первинний ключ зазвичай реалізується як стовпець або група стовпців із забороненими NULL-значеннями. Згідно з вимогами цілісності сутності для ключів сутності створюються унікальні індекси. Основними правилами, за якими призначаються первинні ключі сутності, є такі:

1. первинний ключ повинен однозначно ідентифікувати будь-який екземпляр сутності;
2. первинний ключ по можливості має бути найбільш компактним із усіх потенційних ключів. Як найкращим типом даних для первинного ключа є цілий тип, значеннями якого є множина цілих чисел;
3. первинний ключ може бути складеним, але збільшення кількості стовпців, що входять до нього, не відповідає вимозі компактності. Вимогу компактності також не вдасться виконати, якщо, наприклад, як первинний ключ вибрати занадто широкий стовпець рядкового типу даних;
4. значення первинного ключа не повинні піддаватися частим модифікаціям. Ідеально, якщо бізнес-логіка предметної області така, що ці значення взагалі не передбачається змінювати;
5. правила модифікації первинного ключа повинні контролюватися внутрішньою бізнес-логікою предметної області, а не рішеннями, які приймаються над нею. Наприклад, у базі даних, що розробляється для пот-

реб кадрового обліку підприємства, для сутності ПРАЦІВНИК не варто вибирати як первинний ключ серію і номер паспорта працівника. Хоча ці дані в принципі і володіють властивостями обов'язковості й унікальності, але їх зміна може бути ініційована працівником, а не адміністрацією підприємства;

6. якщо серед інформації, що зібрана про сутність, не вдається виділити дані, що задовольняють перераховані вище вимоги, то рекомендується розглянути можливість створення сурогатного первинного ключа, який, не несучи ніякого семантичного навантаження, просто служить ідентифікатором конкретного екземпляра сутності. Як сурогатний первинний ключ вибираються будь-які коди або ідентифікатори. Зазвичай сурогатний первинний ключ прихований від кінцевого користувача. Прикладом сурогатного первинного ключа є табельний номер працівника, який подається у вихідних формах і відомий працівникам.

Зв'язки без права передачі

Екземпляр сутності в якийсь момент часу пов'язаний з одним екземпляром іншої сутності, а потім цей зв'язок може бути розірваний. Після цього зв’язок можливо встановити між іншими екземплярами сутності. Прикладом такої ситуації служить перехід працівника в інший відділ або студента в іншу групу. Якщо подібне неприпустимо, то такий зв'язок називають зв'язком без права передачі. Зв'язком без права передачі є, наприклад, зв'язок між сутностями «ОСОБА» і «ПАСПОРТ», оскільки паспорт не можна передати іншій людині, а будь-яка спроба зміни паспорта веде до знищення старого і видачі нового.

Іншим варіантом зв'язку без права передачі є зв'язок між сутністю в тому випадку, коли потрібно зберігати відомості про час створення та розриву зв'язку між конкретними екземплярами сутності. Так, якщо для зв'язку між сутностями СТУДЕНТ і ГРУПА треба зберігати інформацію про момент часу, коли студент змінив групу, то такий зв'язок можна трактувати як зв'язок без права передачі.

При моделюванні бази даних існує ряд правил, що стосуються права передачі зв'язку:

1. зовнішній ключ, що реалізовує зв'язок із правом передачі, не слід робити частиною первинного ключа, оскільки такий ключ буде нестабільним при передачі зв'язку;

2. немає сенсу відстежувати історію змін даних, об'єднаних зв'язком

без права передачі, оскільки кожна зміна зводиться до знищення старого екземпляра і створення нового;

3. дві сутності, що об'єднані зв'язком «один-до-одного» з правом передачі, ніколи не об'єднуються в одну.

Принципи побудови діаграм «сутність –зв'язок»

Як уже визначалось, інфологічну модель реляційної бази даних зазвичай подають у вигляді ER-діаграм у нотації Чена. На ER-діаграмах сутність позначається прямокутником, що містить ім'я сутності, а зв'язок – ромбом, від якого проведені лінії до кожної із сутності, що бере участь у зв'язку. Кінці ліній виділяються символами, що задають ступінь й обов'язковість зв'язку.

Розрізняють ER-діаграми для екземплярів сутностей і ER-діаграми для класів сутностей (далі – ER-діаграми).

Діаграми ER-екземплярів є детальною формою зображення зв'язків між сутностями. На цих діаграмах точками або квадратами зображуються всі допустимі значення ключових атрибутів сутності, що вступила у зв'язок, а за допомогою ліній показані можливі зв'язки між точками.

Приклад діаграми ER-екземплярів для бази даних *Викладач читає дисципліну* наведений на рис. 1.28.

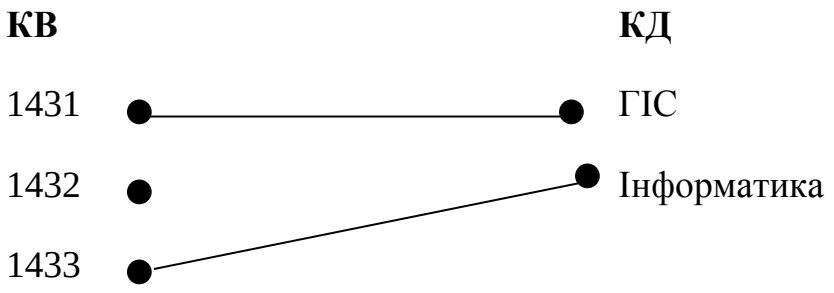


Рис. 1.28 – Приклад подання діаграми ER-екземплярів

Тут: **КВ** (код викладача), **КД** (код дисципліни) – атрибути, які є первинними ключами сутностей ВИКЛАДАЧ і ДИСЦИПЛІНА. На рис. 1.28 показано, що один викладач читає одну дисципліну, але є викладачі, які не читають жодної дисципліни.

Діаграми ER-екземплярів є громіздкими, тому вони застосовуються в тих випадках, коли необхідно визначити правила переходу до моделей нижчих рівнів, наприклад до моделей відношень.

На ER-діаграмах сутність позначається прямокутником, що містить ім'я сутності, а зв'язок – ромбом, від якого проведені лінії до кожної із сутності, що бере участь у зв'язку. Числа над лініями означають кардинальність зв'язку. Клас приналежності (модальність) зв'язку для кожної сутності показується наявністю прямокутної рамки навколо точки. Під прямокутниками вказується ключовий атрибут сутності (рис. 1.29).

Ключ зв’язку **<КВ, КТ>** складений із двох атрибутів. На першому етапі необхідні тільки атрибути, які є ключами сутностей. Інші атрибути та їх функціональні залежності додаються до відношень на більш пізніх етапах.

Рис. 1.29 – Приклад ER-діаграми в нотації Чена

У даному прикладі показано зв’язок типу «один-до-одного» (один викладач читає тільки одну дисципліну). Для сутності ВИКЛАДАЧ зв’язок є необов’язковим (викладач може бути відсутній і не читає жодної дисципліни), а для сутності ДИСЦИПЛІНА – обов’язковим (дисципліну викладачі повинні читати обов’язково). Тому на стороні сутності ДИСЦИПЛІНА точка виділена прямокутною рамкою.

Питання для самодіагностики

1. Охарактеризуйте поняття «атрибут» реляційної БД.
2. Охарактеризуйте поняття «відношення» реляційної БД. Які види відношень ви знаєте?
3. Охарактеризуйте поняття «тип даних» реляційної БД.
4. Охарактеризуйте поняття «домен» реляційної БД.
5. Охарактеризуйте поняття «кортеж» реляційної БД.
6. Охарактеризуйте поняття «таблиця» реляційної БД.
7. Дайте визначення поняттю «первинний ключ». Наведіть приклади.
8. Дайте визначення поняттю «вторинний ключ». Наведіть приклади.
9. Охарактеризуйте всі види зв′язків між таблицями.

РОЗДІЛ 2
ПРОЕКТУВАННЯ РЕЛЯЦІЙНИХ БАЗ ДАНИХ

У даному розділі розглядаються: принципи, методи та етапи проектування реляційних баз даних. Наведено приклад проектування бази даних «Кадастр» для вирішення прикладної задачі обліку нерухомого майна.

2.1 Принципи та теоретичні основи проектування реляційних баз даних

Принципи проектування реляційних баз даних

При проектуванні бази даних вирішуються дві основні проблеми:

1. **Проблема логічного проектування баз даних**, яка пов’язана з відображенням об'єктів предметної області в абстрактні об'єкти моделі даних таким чином, щоб це відображення не суперечило семантиці предметної області і було по можливості кращим (ефективним, зручним і так далі).

2. **Проблема фізичного проектування баз даних**, яка полягає в тому, щоб забезпечити ефективність виконання запитів до бази даних шляхом створення додаткових структур (наприклад, індексів) та використання інших особливостей СКБД.

У даному розділі обмежимося питаннями логічного проектування реляційних баз даних, а конкретні рекомендації щодо фізичного проектування будемо розглядати в методиці створення таблиць засобами СКБД Access.

Основна мета проектування бази даних – це скорочення надмірності даних, що зберігаються, а отже, економія об'єму використовуваної пам'яті, зменшення витрат на багатократні операції оновлення надмірних копій і усунення можливості виникнення суперечностей через зберігання в різних місцях відомостей про один і той же об'єкт.

Таким чином, вирішення проблеми логічного проектування пов’язано з розв’язанням задач *мінімізації надмірності даних* та виключенням можливих *суперечностей збережених даних*.

Указані задачі є досить складними, але принцип їх розв’язання формулюється досить просто: «*Кожен факт повинен з'являтися лише в одному місці (відношенні) бази даних*». Практична реалізація цього принципу базується на методології нормалізації відношень. У результаті такого під-

ходу можна отримати обґрунтоване прийняття рішень про такі питання:

- з яких відношень повинна складатися база даних;
- які атрибути мають бути в цих відношеннях;
- які мають бути зв'язки між відношеннями у схемі бази даних.

При проектуванні реляційних баз даних широке застосування знайшли два підходи:

1. реляційний (класичний) підхід за принципами нормалізації;
2. інфологічне моделювання за принципами методології «сутність–зв'язок».

При реляційному підході етап інфологічного проектування не виконується. Проектування проводиться в термінах реляційної моделі даних методом послідовних наближень до задовільного набору схем відношень. Початковою точкою є подання предметної області у вигляді одного або декількох відношень, і на кожному кроці проектування створюється деякий набір схем відношень, що мають кращі властивості. Процесом проектування є процес нормалізації схем відношень, причому кожна наступна нормальна форма має кращі властивості, ніж попередня.

Кожній нормальній формі відповідає деякий певний набір обмежень. Відношення знаходиться в деякій нормальній формі, якщо задовольняє певний набір обмежень. Прикладом набору обмежень є обмеження першої нормальної форми – значення всіх атрибутів відношення є атомарними. Оскільки вимога першої нормальної форми є базовою вимогою класичної реляційної моделі даних, вважатимемо, що початковий набір відношень уже відповідає цій вимозі.

Реляційна модель даних достатня для моделювання предметних областей різного призначення. Проте проектування реляційної бази даних із використанням принципів нормалізації відношень є дуже складним і незручним, оскільки проектування виконується вручну.

Реляційна модель даних має обмеженість у таких аспектах:

1. Модель не надає достатніх засобів для представлення сенсу даних. Семантика реальної предметної області повинна незалежним від моделі способом уявлятися проектувальником. Зокрема, це відноситься до проблеми подання обмежень цілісності.

2. Хоча весь процес проектування відбувається на основі врахування залежностей, реляційна модель не надає засобів для їх подання.

Незважаючи на це, реляційний підхід до проектування бази даних

доцільно використовувати в тих випадках, коли необхідно прискорити процес проектування і впровадження системи в експлуатацію. Скориставшись реляційним підходом, можна спроектувати оптимальну логічну модель бази даних. Така модель бази даних не має аномалій, пов'язаних із модифікацією бази даних, тобто проблем, що можуть виникнути внаслідок замін, вставок і вилучення даних. Під аномаліями розуміють відхилення від норм, які можуть призвести до порушення посилкової цілісності бази даних чи до виникнення протиріч і неузгодженості даних.

Відсутність невиправданої надмірності даних

Формальне визначення терміну «надмірність бази даних» дати неможливо. Але є загально прийняті трактування стандартних ситуацій, які пов'язані з надмірністю даних. Перша типова ситуація пов’язана з визначенням первинних ключів відношень. Так, не раціонально ідентифікувати кортежі відношень первинними ключами з доменів, що містять текстові дані, тобто недоцільно як первинні ключі використовувати текстові дані. Друга типова ситуація виникає, коли одні й ті ж факти можна отримувати з різних об'єктів бази даних.

Наявність надмірності може принести або користь, або бути недоліком з погляду загальної продуктивності системи. Користь від внесення надмірності до бази даних полягає в можливості прискорення виконання запитів, оскільки частина потрібної інформації вже збережена в готовому вигляді, і не потрібно розшукувати її по різних таблицях, здійснюючи досить складні з'єднання таблиць та обчислення. Але за все потрібно платити – надмірність має бути контрольованою. Тобто необхідна програмна реалізація перевірок того, що надмірні і базові дані адекватно узгоджені між собою.

Ще одним прикладом надмірності є наявність у базі даних обчислюваних атрибутів та пов'язаних із ними обмежень. Обчислюваними називаються атрибути, значення яких можуть бути обчислені на підставі значень базових атрибутів, що зберігаються в базі даних.

З погляду способів реалізації важливо, чи обчислюються похідні атрибути за даними одного запису однієї таблиці, або для їх обчислення потрібно використовувати дані з різних записів або таблиць. У першому випадку можна в таблиці створити обчислюваний стовпець, а в другому випадку потрібна спеціальна процедура. Наприклад, у таблиці є стовпець із датою народження. Щоб дізнатися про поточний вік людини (кількість повних ро-

ків), можна створити обчислюваний стовпець. Якщо в накладній потрібно встановити взаємозв'язок атрибутів «Ціна × Кількість = Сума», то швидше за все знадобиться процедура, оскільки значення ціни навряд чи зберігається в тій же самій таблиці, де зберігаються дані про товарні позиції накладної.

При інфологічному моделюванні за принципами методології «сутність–зв’язок» спочатку виконується проектування інфолологічної моделі. У результаті проектування будується концептуальна схема бази даних із використанням ER-діаграм. Далі здійснюється відображення ER-моделі на реляційну модель за формальними правилами.

Теоретичні основи проектування реляційних баз даних

Основні поняття теорії нормалізації

Введемо поняття: нормалізація, функціональна залежність атрибутів, декомпозиція, принцип нормалізації та типи нормальних форм.

Нормалізація відношень – це ітераційний зворотний процес розбиття початкового відношення на кілька простіших відношень меншої розмірності. Під зворотністю процесу розуміють те, що операція об'єднання отриманих нових відношень має дати початкове відношення без втрат інформації.

Функціональна залежність атрибутів. У відношенні R(A, B) атрибут B залежить від атрибута A (A → B) тоді, коли в кожний момент часу одному й тому самому значенню A відповідає одне значення B.

Декомпозиція та принцип нормалізації. В основу нормалізації покладено принцип декомпозиції. Декомпозиція – це процес розбиття одного відношення на кілька простіших відношень меншої розмірності шляхом застосування до початкового відношення операції проекції реляційної алгебри. У результаті використання операції проекції до існуючого відношення отримаємо нове відношення шляхом вибору певних стовпців із поточного відношення. Якщо результат виконання операції проекція містить кортежі, що повторюються, то залишається тільки один кортеж із групи, що повторюється.

Отриманий у результаті нормалізації склад атрибутів відношень бази даних повинен відповідати таким вимогам: між атрибутами не має бути небажаних функціональних залежностей, групування атрибутів має забезпечувати мінімальне дублювання даних, їх обробку й поновлення без ускладнень і аномалій. Водночас отримані в результаті декомпозиції відношення

не повинні втратити функціональних залежностей початкового відношення, бо це може призвести до спотворення семантики даного відношення.

Аномалії ненормалізованого відношення

Структура реляційних відношень у нормалізованій базі даних має бути оптимальною, тобто такою, яка є найбільш стійкою при внесенні змін у дані та зв'язки між ними. У ненормалізованому відношенні можуть виникнути аномалії, які пов'язані з виконанням операцій підтримки його в актуальному стані. Виявити фактичний прояв тієї чи іншої аномалії можна, лише врахувавши конкретну семантику даних. Проілюструємо це на прикладі конкретного відношення:

КОМПАНІЯ (**Код_працівника**, НомерВідділу, Керівник,

ТипКонтракту).

Аномалія оновлення. Заміна керівника призведе до необхідності внесення змін і модифікацій за кожним працівником даного відділу. Для цього у відношенні КОМПАНІЯ потрібно знайти всі записи про керівника, і в кожній з них узгодженим чином змінити дані про керівника. Таким чином, для підтримання узгодженості даних необхідно виконати цілий ряд змін, що може розглядатись як аномалія, оскільки характер самої зміни повинен стосуватися лише одного певного запису бази даних.

Аномалія поповнення. Приймаючи на роботу нового працівника, необхідно вносити в базу даних відомості не лише про нього, а й про керівника та тип контракту. Це можна розглядати як аномалію, тому що не завжди можливо внести в базу даних відомості про відділ і контракт, оскільки часто працівників беруть на роботу з випробувальним терміном і лише після його проходження підписують певний вид контракту та визначають відділ, де він працюватиме. Таким чином, доведеться створювати запис із NULL-значеннями в полях НомерВідділу та ТипКонтракту.

Аномалія видалення. Припустимо, що звільнились усі працівники певного відділу, тоді разом з інформацією про них у базі даних буде втрачено також інформацію про цей відділ. Якщо ж інформацію про відділ необхідно зберігати якийсь час, то це також може розглядатись як аномалія.

Аномалія надлишковості. Відомості про керівника відділу та тип контракту повторюються в ряді кортежів бази даних. Основні проблеми зі зберіганням надлишку інформації пов'язані не лише з неефективним використанням пам'яті, а й із забезпеченням підтримки узгодженості цих даних.

Описані проблеми пов'язані з поганою нормалізацією відношень.

Можливість виникнення аномалій у процесі модифікації ненормалізованих відношень свідчить про неадекватність моделі даних предметній області. Це означає, що логічна модель даних або просто неправильна, або необхідні додаткові зусилля для реалізації всіх обмежень, що визначені в предметній області. Так, для підтримки відношення КОМПАНІЯ у цілісному стані доведеться витратити додаткові зусилля на створення спеціальних процедур, витрачати дисковий простір на зберігання надмірних даних та неминуче вводити NULL-значення.

Усі ці проблеми можуть бути вирішені ще на стадії логічного проектування, якщо застосовувати спеціальні методики нормалізації реляційної бази даних. Для усунення аномалій оновлення застосовується метод нормалізації відношень, заснований на поняттях функціональної залежності (ФЗ) і нормальних форм (НФ).

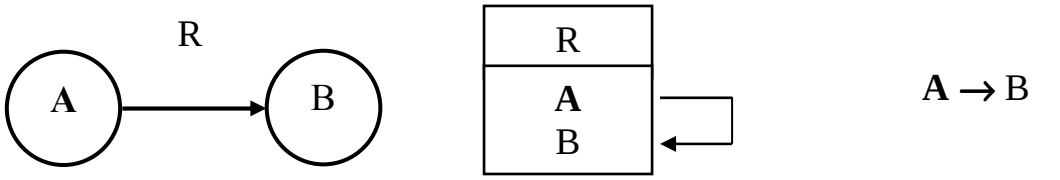
Функціональні залежності атрибутів у відношенні

Кожне відношення БД уміщує як структурну, так і семантичну інформацію. Структурна інформація задається схемою відношення, а семантична виражає функціональні зв'язки між атрибутами.

У теорії нормалізації визначені такі зв'язки між атрибутами: функціональна залежність, тривіальна, нетривіальна та багатозначна залежності, повна функціональна залежність і транзитивна залежність.

Функціональна залежність (ФЗ). Нехай R – відношення. Множина атрибутів Y, яка належить R ($Y \in R$), функціонально залежна від множини атрибутів X ($X \in R$) тоді і лише тоді, коли для будь-якого стану відношення R у всіх кортежах, що мають однакові значення атрибутів X, значення атрибутів Y також збігаються.

Форми наочного подання функціональних залежностей показані на рис. 2.1. Множина атрибутів X називається аргументом функціональної залежності, а множина атрибутів Y – залежною частиною.



а) граф б) схема в) символьна

Рис. 2.1 – Форми подання ФЗ

Означення 1. Функціональна залежність – це така залежність між двома атрибутами (множинами атрибутів) *A* і *B*, коли у будь-який момент часу для будь-якого значення атрибута *A* існує одне і лише одне значення атрибута *B*. Атрибут *A* може бути як атомарним, так і складеним. Цій залежності відповідає співвідношення 1:1 між атрибутами, наприклад, КодПрацівника → Прізвище, КодДисципліни → Назва.

Слід зауважити, що ФЗ не можуть бути виведені із зовнішнього вигляду відношення в конкретний момент часу. Вони відображають взаємозв'язки між об'єктами предметної області, а також є додатковими обмеженнями, що визначені предметною областю. Таким чином, ФЗ – це семантичне поняття. ФЗ виникає, коли за значеннями одних даних у предметній області можна однозначно визначити значення інших даних.

Тривіальна, нетривіальна та багатозначна залежності

Означення 2. Функціональна залежність $X \rightarrow Y$ є *тривіальною*, якщо *X* також функціонально залежить від *Y*, тобто $Y \rightarrow X$.

Означення 3. Залежність $X \rightarrow Y$ є *нетривіальною*, якщо *X* функціонально не залежить від *Y*.

Багатозначна залежність є різновидом функціональної залежності, їй відповідає співвідношення 1 : Б між атрибутами.

Означення 4. У відношенні *R* (*A*, *B*, *C*) атрибут *A* багатозначно визначає атрибут *B*, якщо *B* залежить лише від *A* при будь-яких комбінаціях *A* з іншими атрибутами відношення *R*. Цю залежність позначають так: А –» В.

Приклад багатозначної залежності.

Нехай задано відношення

ПРАЦІВНИК (**КодПрацівника**, Прізвище, ДержавнаНагорода).

У цьому відношенні є багатозначна залежність:

КодПрацівника –» Державна нагорода.

Повна функціональна залежність

Якщо така залежність відношення має складені ключі, то залежність неключових атрибутів від такого ключа може бути повною або частковою.

Означення 5. Атрибут перебуває в повній функціональній залежності, якщо він залежить від всього ключа і не залежить від його складових частин.

Приклад. Маємо відношення *R* (***A, B***, *C*, *D*), ключ якого складається із двох атрибутів *A* і *B*, тобто є складеним (ключові атрибути виділені).

Функціональні залежності у відношенні *R* показані на рис. 2.2.

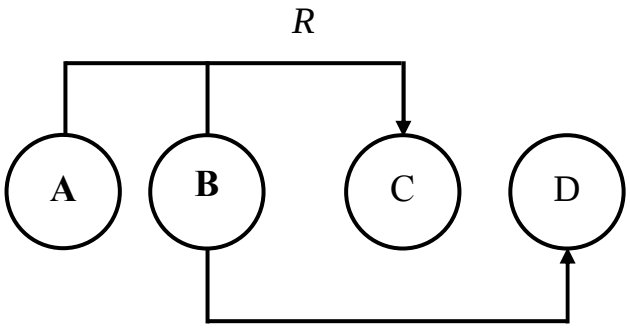


Рис. 2.2 – Функціональні залежності відношення *R* (***A, B***, *C*, *D*)

Атрибут *C* перебуває в повній функціональній залежності, оскільки залежить від усього складеного ключа ***A, B***, а атрибут *D* – у неповній, оскільки залежить лише від його складової частини – атрибута ***B***.

Транзитивна залежність

Транзитивна залежність – це залежність між неключовими атрибутами. Нехай є відношення *R*(***A***, *B*, *C*), у якому атрибут *D* безпосередньо не залежить від ключового атрибута ***A***, а залежить від неключового атрибута *C*, який, у свою чергу, залежить від ***A*** (рис. 2.3). Тоді атрибут *C* транзитивно залежить від ***A*** (пунктирна лінія).

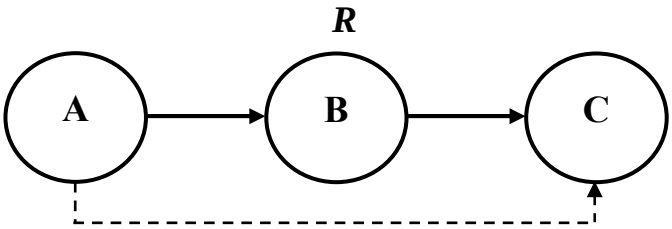


Рис. 2.3 – Схема транзитивної залежності атрибута *C* від ***A***

Неповні та транзитивні залежності вилучають за допомогою декомпозиції відношення на два чи більше інших відношень, які не містять цих залежностей і об'єднання яких дасть початкове відношення.

Правила виведення. Мінімальне покриття

Одна із проблем, пов'язана із застосуванням ФЗ, полягає в тому, що зазвичай при використанні формального визначення вдається виявити значно більше ФЗ, ніж потрібно для коректної нормалізації. З цієї причини залишають мінімально можливий набір ФЗ, який називається мінімальним покриттям.

Мінімальне покриття забезпечує коректне проектування бази даних. Вибір ФЗ для мінімального покриття заснований на використанні такого апарату реляційної алгебри, як правила виведення ФЗ. Розглянемо прості правила виводу, що дозволяють видаляти надлишкові ФЗ:

1. Транзитивність ФЗ (рис. 2.3). Якщо $A \to B$, $B \to C$, то $A \to C$.
2. Додавання ФЗ. Якщо $A \to B$, то A, $C \to B$.
3. Об'єднання ФЗ. Якщо $A \to B$ і $A \to C$, то $A \to B$, C.
4. Декомпозиція ФЗ. Якщо $A \to B$, C, то $A \to B$ і $A \to C$.
5. Псевдотранзитивність ФЗ. Якщо $A \to B$ і B, $D \to C$, то A, $D \to C$.

Графічно правила виводу 2 – 5 проілюстровані на рис. 2.4.

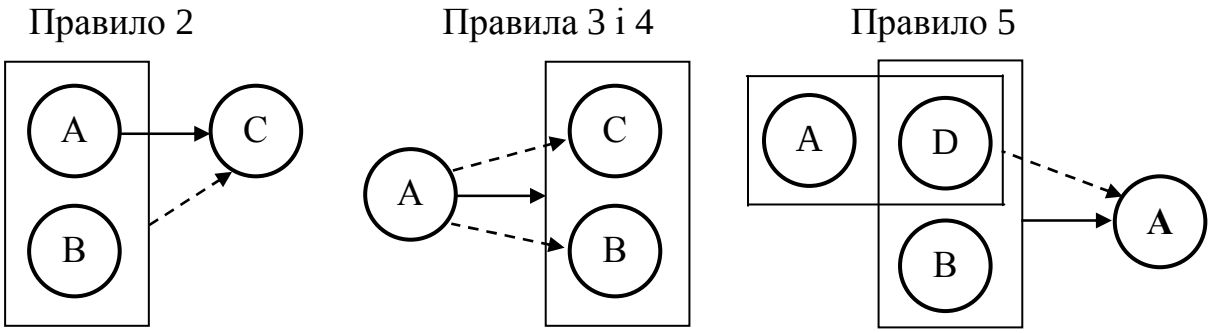


Рис. 2.4 – Графічне подання правил виводу ФЗ

Для правила 1 (рис. 2.3) та правил 2 і 5 пунктиром показані надмірні ФЗ, що підлягають видаленню. Правила 3 і 4 є взаємно зворотними й дозволяють вибрати одну із комбінацій ФЗ (суцільну стрілку або дві пунктирні).

Нормальні форми відношень

Перша нормальна форма (1НФ)

Першим кроком нормалізації є приведення відношення до першої нормальної форми. Відношення в 1НФ повинно відповідати таким вимогам:

- усі атрибути відношення повинні бути унікальними, тобто не допускається їхнього дублювання, а також атомарними, тобто неподільними;
- усі рядки таблиці повинні мати однакову структуру;
- імена стовпців повинні бути різними, а значення однорідними (однакового типу і формату);
- порядок рядків у таблиці не істотний.

Таким чином, будь-яка нормалізована таблиця знаходиться в 1НФ.

Розглянемо приклад зведення відношення до 1НФ:

МАТЕРІАЛ (**КодМатеріалу**, Назва, Характеристика: тип, сорт, розмір).

У цьому відношенні атрибут «Характеристика» є неатомарним, тому потрібно позбутися складеного атрибута й перетворити його в три атомарних атрибути. В результаті зведення до 1НФ відношення матиме вигляд:

МАТЕРІАЛ (**КодМатеріалу**, Назва, Тип, Сорт, Розмір).

Друга нормальна форма (2НФ)

Відношення знаходиться у другій нормальній формі тоді і тільки тоді, коли відношення перебуває в 1НФ і всі його неключові атрибути функціонально повно залежать від первинного ключа. **Неключовий атрибут** – це атрибут, що не входить до складу первинного, або потенційного, ключа.

Якщо відношення має неповні функціональні залежності, то виконують його декомпозицію на два чи більше відношень, які не мають неповних функціональних залежностей і об'єднання яких дасть початкове відношення. Виконавши декомпозицію відношення *R* (див. рис. 2.2), отримаємо два відношення *R*1 і *R*2, які будуть перебувати у 2НФ (рис. 2.5).

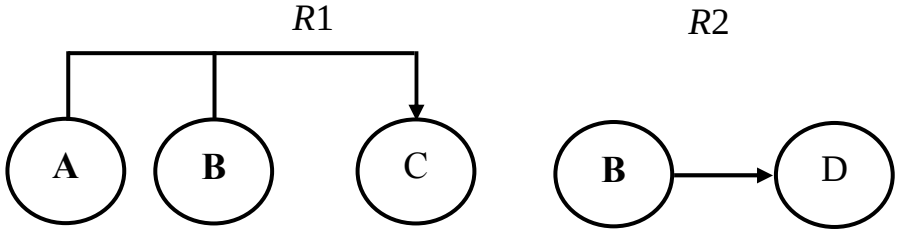


Рис. 2.5 – Функціональні залежності відношень *R*1(***A, B***, *C*) і *R*2(**B**, *D*)

У відношенні *R*1 є тільки один неключовий атрибут *C*, який функціонально повно залежить від складеного первинного ключа. Відношення *R*2 перебуває у 2НФ з тривіальної причини – воно має атомарний ключ, від якого залежить атрибут *D*.

Переваги 2НФ: зручність внесення змін у базу даних. 2НФ повністю виключає можливість виникнення протиріччя даних, а також економить пам'ять при зберіганні відношень у пам'яті ЕОМ. Відношення у 2НФ потрібно аналізувати на присутність транзитивних залежностей.

Третя нормальна форма (3НФ)

Відношення знаходиться у третій нормальній формі тоді і тільки тоді, коли воно перебуває у 2НФ і всі його неключові атрибути взаємно функціонально незалежні, тобто кожний неключовий атрибут не має транзитивної залежності від первинного ключа.

Наприклад, відношення

ВИКЛАДАЧ (**КодВикладача**, Прізвище, Посада, Оклад, КодКафедри, Телефон)

перебуває у 2НФ, але вміщує транзитивну залежність (рис. 2.6).

Рис. 2.6 – Функціональні залежності відношення ВИКЛАДАЧ

У результаті зведення до ЗНФ одержуємо два відношення:

ВИКЛАДАЧ (**КодВикладача**, Прізвище, Посада, Оклад, КодКафедри);

КАФЕДРА (**КодКафедри**, телефон кафедри).

Переваги ЗНФ: виключається надлишкове дублювання інформації про телефон для викладачів однієї кафедри, спрощується процес внесення змін, оскільки у випадку зміни телефону будь-якої кафедри потрібно внести зміни лише в один відповідний запис, а не по всіх викладачах кафедри. Крім того, якщо викладач буде працювати на іншій кафедрі, необхідно змінювати лише код кафедри і не потрібно змінювати номер телефону, що довелося б робити, якби відношення зберігалося в ненормалізованому виді.

Тому можна зробити ще такий висновок: відношення знаходиться в ЗНФ, якщо зміна значення будь-якого його атрибута (крім тих, що входять у первинний ключ) не призведе до необхідності зміни значень інших полів.

Нормальна форма Бойса – Кодда (НФБК)

Нормальна форма Бойса–Кодда – це підсилена ЗНФ, у якій вивчаються залежності ключових атрибутів від неключових.

Відношення перебуває в НФБК, якщо воно перебуває в ЗНФ і в ньому відсутні залежності ключових атрибутів від неключових атрибутів.

Приклад. Нехай задано відношення:

АВТОМОБІЛЬ **(Модель**, КількістьЦиліндрів, КраїнаВиробник).

Залежності атрибутів у цьому відношенні показані на рис. 2.7.

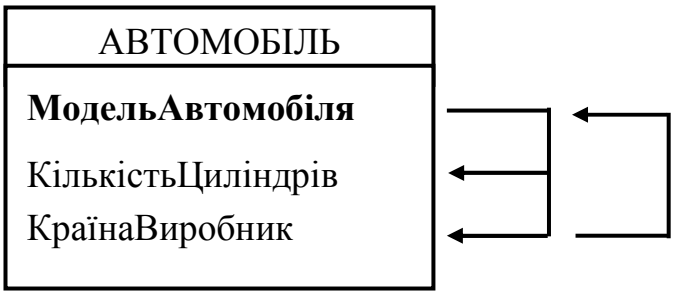


Рис. 2.7 – Функціональні залежності відношення АВТОМОБІЛЬ

Ключовий атрибут **МодельАвтомобіля** залежить від неключового атрибута КраїнаВиробник. Тому при зведенні до БКНФ початкове відношення потрібно розбити на такі два відношення:

АВТОМОБІЛЬ (**МодельАвтомобіля,** КількістьЦиліндрів);

ВИРОБНИК (**КраїнаВиробник**, МодельАвтомобіля).

Зазначимо, що відношення, яке перебуває в НФБК, завжди є відношенням у ЗНФ. Але, навпаки, відношення в ЗНФ не завжди можна привести до нормальної форми Бойса–Кодда, не втративши залежності між його атрибутами.

Приклад. Нехай задано відношення:

ПОШТА (**Місто**, Адреса, Індекс).

Залежності атрибутів у цьому відношенні показані на рис. 2.8.

Зведення відношення до ЗНФ дасть два відношення:

ПОШТА1 **(Місто**, Адреса),

ПОШТА2 **(Адреса**, Індекс).

Вони перебувають у ЗНФ, але не є відношеннями в НФБК, оскільки у відношенні ПОШТА існує залежність Індекс → Місто, тобто відношення в ЗНФ не завжди є відношенням у НФБК.

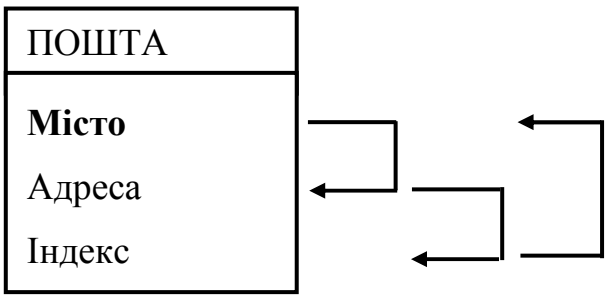


Рис. 2.8 – Функціональні залежності відношення ПОШТА

Якщо ми зведемо це відношення до НФБК, то отримаємо такі відношення:

ПОШТА1 (**Місто**, Адреса),

ПОШТА2 (**Індекс**, Місто).

У цих відношеннях відображені не всі залежності початкового відношення. Так, не відображена залежність Індекс → Адреса, а саме індекс позначає відділення зв'язку, що обслуговує адресатів якоїсь вулиці певного міста.

Таким чином, перехід до НФБК може призвести до втрати важливих для початкового відношення залежностей, а об'єднання отриманих у результаті такої декомпозиції відношень не дасть початкового відношення. Тому при зведенні до НФБК необхідно ретельно вивчати всі залежності і виконувати його лише тоді, коли виконується вимога «об'єднання без втрат».

Нормальна форма Бойса–Кодда має такі самі переваги, що й ЗНФ, тобто виключає можливість виникнення аномалій, пов'язаних із виконанням операцій поповнення, вилучення та заміни даних, а також усуває надлишковість даних.

Зазвичай у базах даних задовольняються приведенням реляційної бази даних до ЗНФ (НФБК). Проте в теорії існують ще четверта і п'ята нормальні форми, які розглянемо на прикладах без строгих визначень.

Четверта нормальна форма (4НФ)

Якщо відношення має багатозначні залежності між атрибутами, то виконують його декомпозицію й отримують 4НФ.

Четверта нормальна форма, будучи достатньо складною, не вважається за обов'язкову на практиці, тому обійдемося при її розгляді без строгих математичних визначень. Розглянемо такий приклад. Нехай потрібно враховувати дані про абітурієнтів, що вступають у ВНЗ.

При аналізі предметної області були виділені такі вимоги:

1. Кожен абітурієнт має право складати іспити на декілька факультетів.
2. Кожен факультет має свій список предметів, що складають абітурієнти.
3. Один і той же предмет може здаватися на декількох факультетах.

Перезалік оцінок, які отримані за однойменними предметами на різних факультетах, не допускається. Абітурієнт зобов'язаний складати всі предмети, що вказані для факультету, на який він вступає.

Припустимо, що в одному відношенні *Іспити* потрібно зберігати дані про предмети, які повинні здавати абітурієнти (табл. 2.1).

Таблиця 2.1 – Відношення *Іспити*

Абітурієнт	**Факультет**	**Предмет**
Іванов	Математичний	Математика
Іванов	Математичний	Інформатика
Іванов	Фізичний	Математика
Іванов	Фізичний	Фізика
Петров	Математичний	Математика
Петров	Математичний	Інформатика

Ключ відношення *Іспити* є складеним і містить усі три його атрибути. У відношенні зберігається інформація про те, що абітурієнт Іванов вступає на два факультети (математичний і фізичний), а абітурієнт Петров – тільки на математичний факультет. На математичному факультеті потрібно складати математику та інформатику, а на фізичному – математику і фізику.

У відношенні *Іспити* є аномалії оновлення, вставки і видалення, які полягають у тому, що жодну з цих операцій не можна виконати для єдиного кортежу відношення без урахування інших його кортежів.

Аномалія додавання. При спробі додати в це відношення новий кортеж, наприклад (Котов, Математичний, Математика), необхідно додати і кортеж (Котов, Математичний, Інформатика), оскільки всі абітурієнти математичного факультету зобов'язані складати однакові предмети.

Аномалія видалення. При спробі видалити кортеж (Іванов, Математичний, Математика) необхідно видалити і кортеж (Іванов, Математичний, Інформатика) з тієї ж самої причини. Якщо видалити кортеж (Іванов, Фізичний, Математика), а також і кортеж (Іванов, Фізичний, Фізика), то буде втрачена інформація про предмети, які повинні здаватися на фізичному факультеті.

Аномалія оновлення. При спробі змінити список іспитів, що здаються на факультеті, потрібно це робити для всіх абітурієнтів.

Таким чином, вставка і видалення кортежів не можуть бути виконані незалежно від інших кортежів відношення.

Для нормалізації відношення *Іспити* не можна використовувати нор-

мальні форми, засновані на ФЗ, оскільки в ньому є взаємозв'язок із багатозначною залежністю. На понятті багатозначної залежності засновано визначення 4НФ.

Відношення *R* знаходиться в 4 НФ тоді і тільки тоді, коли відношення знаходиться в НФБК і не містить нетривіальних багатозначних залежностей, таких, що для багатозначної залежності X –» Y | Z не існує 3 НФ X → Y і X → Z.

Відношення *Іспити* знаходиться в НФБК, але не в 4 НФ. Використовуючи поняття багатозначних залежностей, можна без втрат виконати його декомпозицію на два відношення: ВСТУП(хто куди поступає) і ЕКЗАМЕН (де що здавати). Ці відношення такі:

ВСТУП (**Факультет**, **Абітурієнт**) і

ЕКЗАМЕН (**Факультет**, **Предмет**).

В отриманих відношеннях усунені аномалії вставки і видалення, що характерні для відношення *Іспити*. Отримані відношення знаходяться в 4НФ, залишаючись повністю ключовими, і в них як і раніше немає ФЗ.

Відношення з нетривіальними багатозначними залежностями виникають, як правило, в результаті природного з'єднання двох відношень по загальному полю, яке не є ключовим в жодному із відношень. Фактично це приводить до спроби зберігати в одному відношенні інформацію про дві незалежні сутності. У даному випадку список екзаменів, які здаються на різних факультетах, ніяк не залежить від того, які абітурієнти куди вступають.

Якщо виявиться необхідним зберігати оцінки, отримані при складанні іспитів абітурієнтами, то відношення *Іспити* доведеться залишити в початковому стані, додавши до нього поле *Оцінка*, тому що більше цю оцінку розміщувати ніде. Щоб отримати відношення для зберігання оцінок, потрібно виконати природне з'єднання відношень ВСТУП і ЕКЗАМЕН по полю *Факультет*, повернувшись до відношення *Іспити*. Після цього з'явиться можливість заносити оцінки, які функціонально залежать від складеного ключа відношення *Іспити*.

Як ще один приклад 4НФ можна навести ситуацію, коли співробітник може мати багато місць роботи і багато дітей. Зберігання інформації про місця роботи і дітей в одному відношенні приводить до виникнення нетривіальної багатозначної залежності (Працівник –» Робота | Діти).

Питання для самодіагностики

1. Дайте визначення проблемам і принципам проектування реляційних баз даних.
2. Що таке оптимальна логічна модель даних?
3. Що таке аномалія та якими аномаліями характеризується ненормалізоване відношення?
4. Дайте визначення функціональній залежності.
5. Дайте визначення неповній функціональній залежності.
6. Що таке транзитивна залежність?
7. Дайте визначення багатозначній залежності.
8. У чому полягає суть зведення реляційного відношення до 1НФ?
9. У чому полягає суть зведення реляційного відношення до 2НФ?
10. У чому полягає суть зведення реляційного відношення до ЗНФ?
11. У чому полягає суть зведення реляційного відношення до 4НФ?
12. У чому полягає суть зведення реляційного відношення до НФБК?
13. Охарактеризуйте переваги нормалізованої бази даних.

Практичні завдання для самодіагностики

Завдання 1. У проекті використовуються деталі, які постачаються кількома постачальниками. Кожний постачальник постачає деталі лише певного виду, тобто неможлива поставка одним постачальником деталей різного виду. Кожний проект має свого керівника та виконавців, закріплених за одним проектом.

Потрібно спроектувати реляційну базу даних, що характеризує описану предметну область, і звести її до ЗНФ (4НФ).

Завдання 2. Наведено реляційне відношення, що перебуває в 1НФ:

ПОСТАВКА (код постачальника, назва постачальника, адреса постачальника, телефон, номер розрахункового рахунку постачальника, код товару, назва товару, одиниця вимірювання, оптова ціна за одиницю, номер договору на постачання, дата укладання договору, дата виконання договору, кількість товару згідно договору.

Потрібно побудувати діаграму функціональних залежностей та звести відношення до ЗНФ (4НФ).

Завдання 3. Дано реляційне відношення, що перебуває в ІНФ:

УСПІШНІСТЬ (прізвище, ім'я та по батькові студента, номер залікової книжки, курс, група, код і назва спеціальності, код предмета, назва пре-

дмета, табельний номер, прізвище, ім'я та по батькові викладача, кафедра, де працює викладач, посада викладача, дата здачі екзамену, оцінка).

Потрібно побудувати діаграму функціональних залежностей та звести до ЗНФ (4НФ), урахувавши, що один викладач може читати кілька дисциплін.

Завдання 4. Дано реляційне відношення, що перебуває в 1 НФ:

ДЕТАЛЬ (код деталі, назва деталі, собівартість деталі, код технологічної операції, назва технологічної операції, код і назва обладнання, на якому виконується операція, пооперaційна трудомісткість обробки деталі).

Потрібно побудувати діаграму функціональних залежностей та звести відношення до ЗНФ (4НФ).

2.2 Етапи проектування реляційних баз даних

Проектування інфологічної моделі бази даних

На цьому етапі виконується аналіз предметної області, визначаються мета створення та вимоги до бази даних, основні об’єкти та зв’язки між ними, інформаційні потреби користувача (аналіз запитів), а також уточнюється послідовність дій майбутніх користувачів при роботі з базою даних і задачі, що будуть вирішуватися. На основі отриманої інформації розробляється концептуальна схема бази даних із використанням моделі «сутність–зв’язок».

Загальний підхід до побудови бази даних із використанням ER-методу полягає у виконанні таких дій:

1. Побудова діаграми ER-типу, в яку мають бути включені всі сутності та зв'язки, які є важливими з погляду інтересів організації.
2. Аналіз зв'язків і визначення їх характеристик: ступінь зв'язку та приналежність.
3. Побудова набору попередніх відношень із вказівкою передбачуваного первинного ключа для кожного відношення.
4. Підготовка списку всіх атрибутів і призначення кожного із цих атрибутів одному з відношень. Відношення повинні знаходитися в НФБК.
5. Побудова схеми даних.

Проектування логічної схеми бази даних

На основі розробленої моделі даних визначається структура первинних таблиць бази даних, типи даних і ключові поля кожної таблиці. Вико-

нується нормалізація таблиць і будується логічна схема бази даних, у якій визначаються зв’язки між таблицями.

Правила отримання відношень із ER-діаграм залежать від приналежності та ступеня зв’язку сутностей. Поєднання трьох типів зв'язків із двома типами приналежності дають можливість опису множини різних варіантів зв'язків у предметній області. Розглянемо не всі, а тільки найбільш поширені варіанти зв’язків сутностей, відповідні ER-діаграми та правила побудови на їх основі відношень. Як приклади будемо використовувати правила взаємин між сутностями **ВИКЛАДАЧ** і **ДИСЦИПЛІНА**, що прийняті в різних навчальних закладах.

Правила для бінарних зв’язків 1:1

Правило 1. Якщо ступінь бінарного зв'язку 1:1 і клас приналежності обох сутностей є обов'язковим, то потрібне тільки одне відношення. Первинним ключем цього відношення може бути ключ будь-якої з двох сутностей, наприклад:

ДИСЦИПЛІНА ВИКЛАДАЧА (**КВ**, Прізвище, Посада, КД, Назва, Час).

Відношення гарантує одноразову появу кожного значення КВ і КД. Воно не буде мати порожніх даних та надмірних даних, що повторюються.

Правило 2. Якщо ступінь бінарного зв'язку 1:1 і приналежність однієї сутності є обов'язковою, а іншої – необов'язковою (рис. 2.10), то необхідна побудова двох сутностей. Під кожну сутність виділяється одне відношення. Ключ сутності повинен служити первинним ключем для відповідного відношення. У сутність з обов'язковою приналежністю додається ключ сутності, для якої приналежність є необов'язковою (див. рис. 1.29).

ER-діаграма, де сутність ВИКЛАДАЧ має обов'язкову приналежність, а сутність ДИСЦИПЛІНА – необов'язкову, відображається на два відношення:

ВИКЛАДАЧ (**КВ**, Прізвище, КД);

ДИСЦИПЛІНА (**КД**, Назва, Час).

Відповідні таблиці наведені на рис. 2.10.

Викладачі

КВ	Прізвище	КД
В1	Іванов В. А.	Д1
В2	Петров С. М.	Д2

Дисципліни

КД	Назва	Час
Д1	Математика	54
Д2	Інформатика	72

Рис. 2.10 – Приклад таблиць для правила 2

У відношення ВИКЛАДАЧ і в таблицю ВИКЛАДАЧІ включено додатковий атрибут КД, який є зовнішнім ключем зв’язку.

Правило 3. Якщо ступінь бінарного зв'язку дорівнює 1:1 і приналежність жодної із сутностей не є обов'язковою (рис. 2. 11), то необхідно використовувати три відношення: по одному для кожної сутності і одне відношення для зв'язку. Ключ кожної сутності використовується як первинний ключ відповідного відношення. Відношення зв'язку повинне мати в числі своїх атрибутів ключі кожної сутності.

Рис. 2. 11 – Сутності ВИКЛАДАЧ і ДИСЦИПЛІНА мають необов'язкову приналежність

Наведена ER-діаграма відображається на три відношення:

ВИКЛАДАЧ (**КВ**, Прізвище, КД);

ДИСЦИПЛІНА (**КД**, Назва, Час);

ДИСЦИПЛІНИ ВИКЛАДАЧА (**КД**, **КВ**).

Відповідні таблиці наведені на рис. 2.12. Таблиця ДИСЦИПЛІНИ ВИКЛАДАЧІВ є зв’язковою таблицею.

Викладачі

КВ	Прізвище	КД
В1	Іванов В. А.	Д1
В2	Петров С. М.	Д2

Дисципліни

КД	Назва	Час
Д1	Математика	54
Д2	Інформатика	72

Дисципліни викладачів

КВ	**КД**
В1	Д1
В2	Д2

Рис. 2.12 – Приклад таблиць для правила 3

Правила для бінарних зв’язків 1 : М

Для таких зв’язків використовуються два правила, кожне з яких визначається класом приналежності М-зв′язкової сутності. Приналежність 1-зв'язкової сутності на результат не впливає.

Правило 4. Якщо ступінь бінарного зв'язку дорівнює 1:М і приналежність М-зв’язної сутності є обов'язковою (рис. 2.13), то використовуються два відношення, по одному на кожну сутність. Ключ кожної сутності є первинним ключем для відповідного відношення. Відношення М-зв'язкової

сутності повинне мати в числі своїх атрибутів ключ 1-зв’язкової сутності.

Рис. 2.13 – Сутність ВИКЛАДАЧ має необов'язкову приналежність, а сутність ДИСЦИПЛІНА – обов'язкову

Наведена ER-діаграма відображається на два відношення:

ВИКЛАДАЧ (**КВ**, Прізвище, КД);

ДИСЦИПЛІНА (**КД**, Назва, Час, КВ).

Відповідні таблиці наведені на рис. 2.14.

Викладачі

КВ	Прізвище	КД
В1	Іванов В. А.	Д1
В2	Петров С. М.	Д2

Дисципліни

КД	Назва	Час	КВ
Д1	Математика	54	В1
Д2	Інформатика	72	В2

Рис. 2.14 – Приклад таблиць для правила 4

У відношення та таблицю ДИСЦИПЛІНИ для організації зв’язку додатково включено ключ 1-зв’язкової сутності ВИКЛАДАЧ.

Правило 5. Якщо ступінь бінарного зв'язку рівний 1:M і приналежність M- зв′язкової сутності є необов'язковою (рис. 2.15), то необхідне формування трьох відношень: по одному для кожної сутності і одне відношення для зв'язку. Ключ кожної сутності використовується як первинний ключ відповідного відношення. Відношення зв'язку повинне мати в числі своїх атрибутів ключі кожної сутності.

Рис. 2.15 – Сутності ВИКЛАДАЧ і ДИСЦИПЛІНА мають необов'язкову приналежність

Наведена ER-діаграма відображається на такі ж три відношення та таблиці, як і для правила 3 (див. рис. 2.12).

Правило 6. Якщо ступінь бінарного зв'язку рівний M:M, то для збе-

рігання даних необхідно три відношення: по одному для кожної сутності і одне відношення для зв'язку. Ключ кожної сутності використовується як первинний ключ відповідного відношення. Відношення зв'язку повинне мати в числі своїх атрибутів ключі кожної сутності (див. рис. 2.13).

Проектування фізичної моделі бази даних

Починаючи з даного етапу, всі роботи виконуються в середовищі системи керування базами даних MS Access. На даному етапі виконуються такі роботи:

1. Визначається спосіб створення бази даних – ручний, починаючи з «нуля», або автоматизований – з використанням майстрів MS Access.
2. Створюються структури таблиць: вводяться імена полів, задаються типи даних і властивості полів (розмір поля, формат поля, умови на значення поля та інше), визначаються первинні ключі таблиць.
3. Складається схема даних шляхом встановлення зв'язків між порожніми таблицями. Такий підхід дозволяє спростити внесення змін до структур таблиць у випадку, якщо MS Access виявить порушення цілісності даних. Заповнені таблиці коректувати набагато складніше.
4. Створюються всі об'єкти, які необхідні для роботи з базою даних: запити, форми, звіти, макроси та ін.
5. Виконується тестування бази даних. Перед введенням реальних даних проводиться заповнення таблиць набором спеціально підібраних «правильних» і «помилкових» тестових даних. За допомогою правильних тестових даних перевіряється структура таблиць. Дані з помилками служать для перевірки реакції бази даних на допущені порушення. У процесі тестування бази даних перевіряється правильність виконання запитів та побудови форм і звітів. При необхідності вносяться зміни до бази даних.

Приклад проектування бази даних

Як приклад, на якому буде розглянуто тему «Створення бази даних у СКБД Microsoft Access», розглянемо етапи проектування бази даних типу «*Облік нерухомості та мешканців квартир*». Вибір прикладу обумовлений тим, що подібні бази даних можуть знайти широке застосування при обліку різноманітних об′єктів нерухомості та їх характеристик, починаючи із фізичних осіб, прописаних у квартирах, і закінчуючи різноманітними документами на права власності. У базі даних будуть враховані найбільш істотні об'єкти даної предметної області і характеристики цих об'єктів.

Основне призначення прикладу – показати підхід до проектування та створення реляційних баз даних засобами СКБД Microsoft Access. Отримані знання будуть корисними при розробці власних баз даних для різних предметних областей.

Призначення додатка. Постановка задачі

Одному з муніципальних відділів поставлено завдання з обліку нерухомості, яка знаходиться на балансі міста.

Потрібно розробити базу даних **«Кадастр»**, за допомогою якої буде вестися облік міських будівель, мешканців, нерухомого майна, яким володіють мешканці, та сплати податків. За результатами обліку нерухомого майна працівники муніципальних відділів виконують аналіз змін нерухомості, мешканців та своєчасності сплати налогів.

Основні задачі, які будуть вирішуватися в додатку

Група 1. Задачі обліку міської нерухомості:

- облік міських будинків;
- облік квартир у цих будинках;
- облік мешканців квартир.

Група 2. Задачі обліку нерухомого майна:

- облік нерухомого майна;
- облік сплати податків за нерухоме майно;
- облік боржників.

Група 3. Задачі аналізу змін у нерухомості та мешканцях:

- облік квартиронаймачів;
- облік правоустановчих документів на квартири;
- динаміка щодо забудови міста.

Група 4. Для забезпечення задач обліку й аналізу необхідно передбачити вирішення допоміжної задачі з ведення такої довідкової інформації:

- за районами міста;
- за вулицями міста;
- за матеріалами, з яких споруджено будинки;
- за змінами податків на нерухомість.

Основні вимоги до бази даних

База даних **«Кадастр»** повинна містити набір даних, який є необхідним для вирішення перерахованих задач.

Система керування повинна забезпечувати:

- введення, редагування і перегляд списків районів міста, вулиць міста, матеріалів, із яких споруджено будинки;
- сортування і вибір необхідної інформації за допомогою запитів;
- обчислення кількості будівель по районах та вулицях, а також кількості їх мешканців і відображення узагальненої інформації щодо нерухомості за допомогою запитів;
- обчислення податків на нерухоме майно, а також відображення узагальненої інформації щодо податків за допомогою запитів.

Результати всіх запитів мають бути оформлені у вигляді екранних форм для полегшення ведення БД та аналізу результатів господарської діяльності. Для створення повноцінного додатка і полегшення роботи оператора має бути розроблений інтерфейс користувача за допомогою головної кнопкової форми.

Проектування інфологічної моделі бази даних

При обліку нерухомості основними сутностями є: будівлі, квартири, мешканці та додаткова нерухомість (рис. 2.16).

Будівля	**Квартира**	**Мешканець**	**Додаткова нерухомість**
Код будівлі	**Код квартири**	**Код мешканця**	**Код нерухомості**
Індекс	Код будівлі	Код квартири	Додаткова нерухомість
Місто	Номер квартири	Прізвище	Житлова площа
Номер будівлі	Поверх	Ім’я	Ціна метра^2
Фото будівлі	Кількість кімнат	По батькові	
Наявність ліфту	Загальна площа	Ідентифікаційний код	
Кількість поверхів	Житлова площа	Квартиронаймач	
Номер ЖЕКу	Наявність балкону	Рік народження	
Кадастровий код	Площа балкона	Статус	
Рік будування	Висота стін		
	Тип права		
	Особовий рахунок		

Рис. 2.16 – Сутності обліку нерухомості та їх атрибути

Концептуальна схема бази даних «Кадастр» обліку нерухомості наве-

дена на рис. 2. 17. Між сутностями існують бінарні зв′язки ступеня 1:М. Перехід від сутностей до відношень будемо здійснювати згідно з правилом 4 для всіх пар, у яких приналежність зв′язку М-зв′язної сутності є необов′язковою, а для сутностей **Нерухомість – Мешканець** (приналежність зв′язку сутності **Мешканець** є необов′язковою) – правило 5.

Виходячи з цього, до складу бази даних «Кадастр» включимо такі відношення:

Будівля (**Код_будівлі**, Індекс, Місто, Номер_будівлі, Фото_будівлі, Наявність_ліфту, Кількість_поверхів, Номер_ЖЕКу, Кадастровий_код, Рік_будування);

Квартира (**Код_квартири**, Номер_квартири, Поверх, Кількість_кімнат, Загальна_площа, Житлова_площа, Наявність_балкону, Площа_балкона, Висота_стін, Тип_права, Особовий_рахунок);

Мешканець (**Код_мешканця**, Прізвище, Ім′я, По_батькові, Ідентифікаційний_код, Квартиронаймач, Рік_народження, Статус);

Додаткова нерухомість (**Код_нерухомості**, Назва_нерухомості, Житлова_площа, Ціна_м_кв);

Податки (Код_податку, Сплата_податку, Ставка_податку, Дата_встановлення).

Згідно з правилом 5, крім сутностей **Квартира**, **Мешканець** та **Нерухомість**, створимо відношення, які характеризують будівлі:

Район (**Код_району**, Назва_району);

Вулиця (**Код_вулиці**, Тип_вулиці, Назва_вулиці);

Матеріал (**Код_матеріалу**, Назва_матеріалу);

Документ (**Код_документа**, Вид_договору, Номер_документа, Дата_реєстрації, Номер_у_книзі_записів);

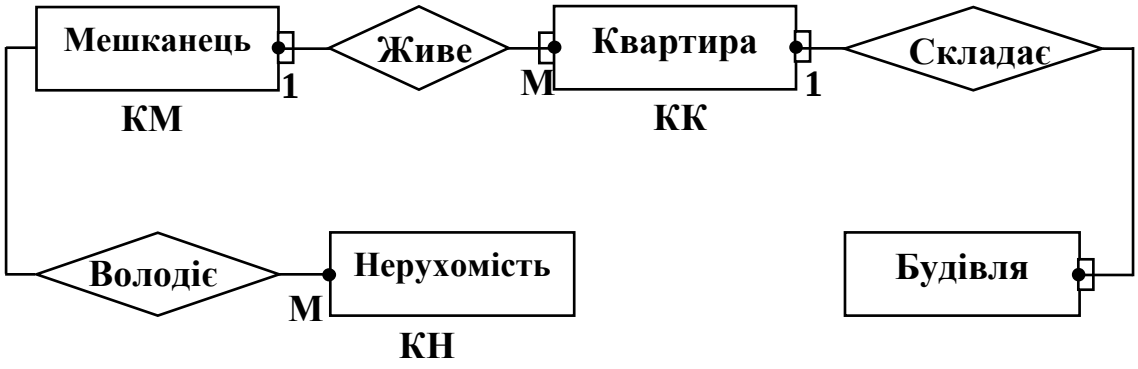


Рис. 2.17 – ER-діаграма обліку нерухомості

У наведених відношеннях атрибути мають такі типи:

Код_будівлі, Індекс, Кількість_поверхів, Номер_ЖЕКу, Рік_будування, Код_квартири, Номер_квартири, Поверх, Кількість_кімнат, Загальна_площа, Житлова_площа, Площа_балкона, Висота_стін, Особовий_рахунок, Код_мешканця, Ідентифікаційний_код, Рік_народження, Код_документа, Номер_у_книзі_записів, Код_вулиці, Код_району, Код_матеріалу, Код_податку, Код_нерухомості, Житлова_площа, Ціна_м_кв – тип **Числовой**;

Дата_реєстрації, Дата_встановлення – тип **Дата/Время**;

Місто, Номер_будівлі, Кадастровий_код, Тип_права, Прізвище, Ім′я, По_батькові, Статус, Вид_договору, Номер_документа, Назва_району, Тип_вулиці, Назва_вулиці, Назва_матеріалу, Додаткова_нерухомість – тип **Текстовый**;

Фото_будівлі – тип **Поле объекта OLE**;

Наявність_ліфту, Наявність_балкону, Квартиронаймач, Сплата_податку – тип **Логический**.

Проектування логічної схеми бази даних

Виконаємо нормалізацію відношень і побудуємо логічну схему бази даних. Відношення ***Будівля*** має атрибути ***Район***, ***Вулиця***, ***Матеріал***, які не залежать від первинного ключа **Код_будівлі**. У результаті декомпозиції цього відношення отримаємо чотири відношення: Будівля, Район, Вулиця, Матеріал.

На основі отриманих відношень створені відповідні таблиці. Логічна схема нормалізованої бази даних **«Кадастр»** наведена на рис. 2.18.

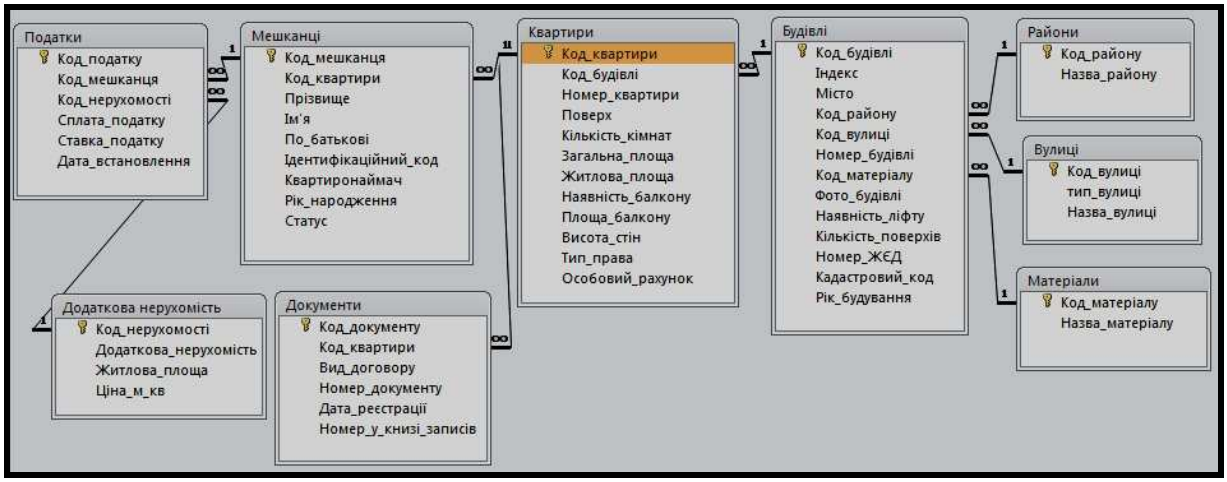


Рис. 2.18 – Логічна схема бази даних «Кадастр»

На цьому завершується етап «Розробка логічної схеми бази даних». Результати проектування бази даних **«Кадастр»** будемо використовувати при вивченні способів створення баз даних за допомогою засобів системи керування базами даних Microsoft Access.

Питання для самодіагностики

1. Охарактеризуйте етапи проектування реляційних баз даних.
2. Що виконується на етапі проектування інфологічної моделі бази даних?
3. Поясніть сутність побудови ER-діаграми за нотацією Чена.
4. Які графічні символи використовуються в нотаціях Чена?
5. У чому полягає алгоритм визначення сутностей?
6. Поясніть сутність проектування логічної схеми бази даних.
7. Навіщо формулюються основні задачі та вимоги до додатку перед проектуванням інфологічної моделі?

РОЗДІЛ 3
СТВОРЕННЯ РЕЛЯЦІЙНИХ БАЗ ДАНИХ З ВИКОРИСТАННЯМ СКБД MICROSOFT ACCESS

У даному розділі розглядаються: прийоми та засоби створення таблиць, запитів, форм і звітів бази даних у середовищі MS Access, публікація баз даних у мережі Internet, методика розробки додатків із використанням макросів і мови програмування Visual Basic for Application (VBA).

Загальні відомості про MS Access

Система керування базами даних Microsoft Access – це програма, призначена для створення і супроводу реляційних баз даних.

База даних MS Access – це сукупність об'єктів (таблиць, форм, запитів та ін.), які використовуються для збереження, перегляду, зміни і виводу даних на друк. MS Access забезпечує введення даних у таблиці бази даних, їх зберігання і супровід, а також отримання із сукупності цієї інформації даних, необхідних для ухвалення важливих рішень.

Усі об'єкти БД зберігаються в одному файлі, який має унікальне ім'я, формат і розширення .mdb у версії MS Access 2003 та .accdb у версії MS Access 2010. Файл нової БД містить приховані системні таблиці, в які заноситься інформація про всі створювані об'єкти БД. У міру заповнення бази даних розмір її файла швидко збільшується, оскільки у файл додаються нові об'єкти і дані, а інформація про об'єкти, що видаляються, зберігається.

Запуск MS Access виконується за командою **Пуск/Все программы/Microsoft Access** головного меню Windows або за допомогою ярлика  , розташованого на робочому столі. Після завантаження MS Access на екрані з'являється його робочий простір (рис. 3.1).

Об'єкти бази даних групуються за типами. У MS Access таких типів п′ять: таблиці, запити, форми, звіти та макроси. У БД об'єкти одного типу повинні мати унікальні імена. Імена об'єктів різних категорій можуть збігатися, наприклад, таблиця ***Будівлі***, форма ***Будівлі***, звіт ***Будівлі*** і т. д. Проте імена таблиць і запитів повинні розрізнятися.

Таблиці складають основу реляційної бази даних і служать для зберігання даних. Решта об'єктів БД пов'язана з таблицями та залежить від них. Таблиця складається із заголовка і тіла. Заголовок включає імена полів, а тіло містить рядки (записи). Кожен запис містить дані про конкрет-

ний екземпляр об'єкта, наприклад інформацію про одного мешканця квартири (його прізвище, ідентифікаційний код і рік народження). При відкритті таблиці в режимі перегляду на екрані можна побачити всі поля та всі записи, що зберігаються в таблиці.

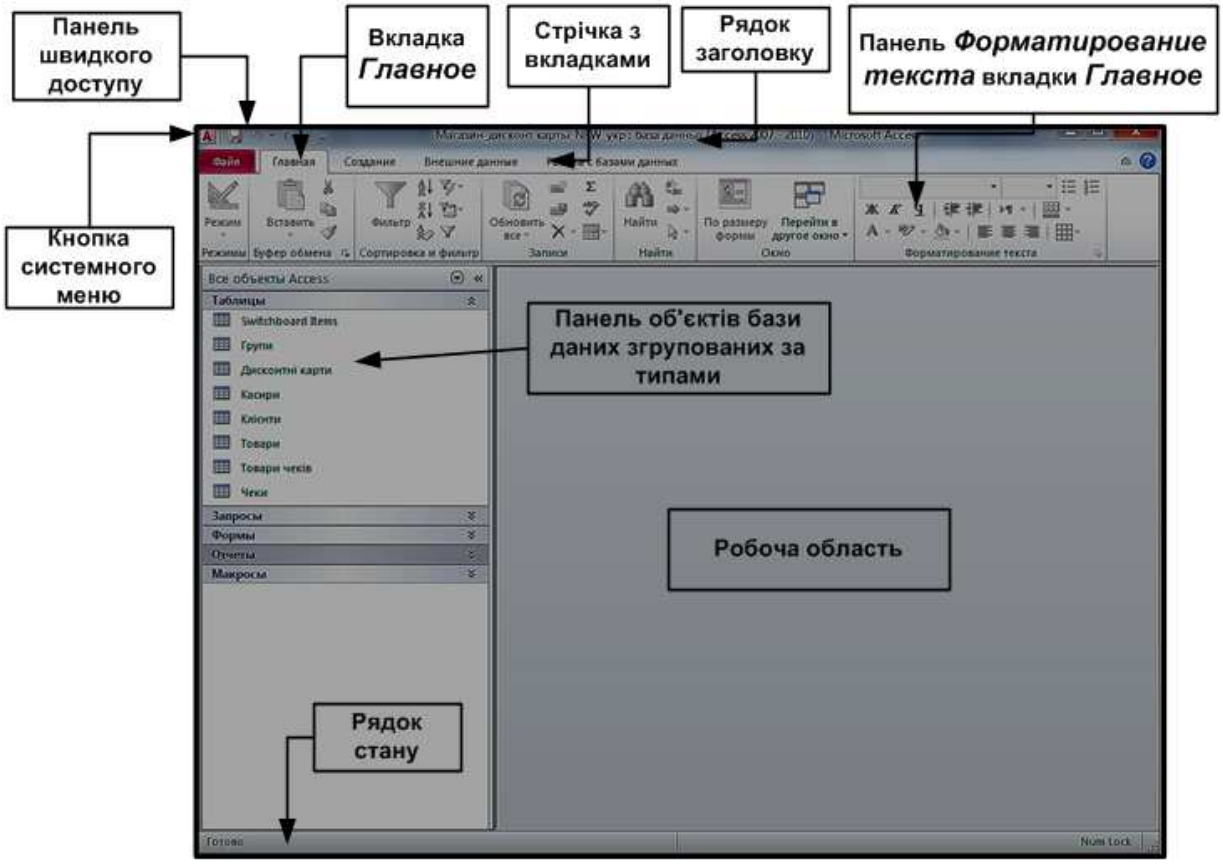


Рис. 3.1 – Робочий простір MS Access

Запити призначені для управління даними і дозволяють вибирати з таблиць БД відомості, які відповідають певному критерію. За допомогою запитів можна автоматизувати оновлення або видалення записів в одній або декількох таблицях, а також виконувати обчислення на підставі значень, що зберігаються в таблиці. Крім того, можна створювати нові таблиці, використовуючи дані однієї або декількох таблиць, які вже існують. У MS Access використовуються запити за зразком QBE і запити, створені користувачем на мові побудови запитів SQL.

Форми використовуються для введення даних, відображення їх на екрані або управління роботою додатка. Форми дозволяють представити дані на екрані в наочній формі, є основним інструментом супроводу баз даних і засобом обмеження доступу до них. За допомогою форми можна

запустити макрос або процедуру, що виконують обробку даних.

Звіти призначені для створення документів, які надалі можуть бути роздруковані або включені в документ іншого застосування. У звіт можуть бути включені результати підсумкових розрахунків та аналізу даних, що наведені у зручному і наочному вигляді, зокрема у вигляді графіків та діаграм.

Макроси – це прості програми, за допомогою яких можна автоматизувати виконання операцій, що часто повторюються, наприклад, відкриття форми, друк певного звіту або вибір команди меню. Використання макросів дозволяє виконувати рутинні процедури простим натисненням кнопок або комбінацій клавіш, а також активізацією спеціальних команд меню.

Модулі – це спеціальні підпрограми, написані на мові Visual Basic для реалізації нестандартних процедур додатка.

У файлі бази даних різні типи об'єктів можуть бути створені за допомогою конструктора або з використанням різних майстрів. Майстри допомагають створювати об'єкти в режимі діалогу, дають підказки користувачеві і пропонують свої рішення.

Дії над об'єктами

У робочому просторі бази даних об'єкти подаються у вигляді піктограм та імен (рис. 3.2). Над об’єктами бази даних можуть виконуватися такі дії: відкриття, закриття, модифікація об'єкта та його видалення.

Об′єкти, що були відкриті, відображаються на екрані у вигляді вікна, наприклад, вікна таблиці, вікна форми. Вікна об'єктів використовуються для перегляду, введення або редагування даних. Відкрити об'єкт можна такими способами: двічі клацнути мишею на піктограмі об'єкта або виділити об'єкт і натиснути клавішу **Enter**. При закритті об'єкта вікно закривається.

Правила та прийоми роботи користувача в MS Access.

Інтерфейс – це засоби і правила, що забезпечують взаємодію користувача з комп'ютером або додатком Windows. У MS Access реалізований стандартний віконний інтерфейс, прийнятий для додатків Windows. Основу інтерфейсу складають вікна. Крім того, використовуються стандартні діалогові та інші вікна Windows. Робота у вікнах MS Access виконується так само, як і у вікнах інших додатків.

Головне вікно MS Access відкривається при запуску додатка. Вікно містить стандартний набір елементів (див. рис. 3.2). Розглянемо їх призначення.

Рядок заголовку. Включає кнопку системного меню, назву додатка і кнопки управління вікном: згорнути, розгорнути й закрити вікно.

Стрічка з вкладками. Складається із стандартного набору вкладок: Файл, Главное, Создание, Внешние данные, Работа с базами данных.

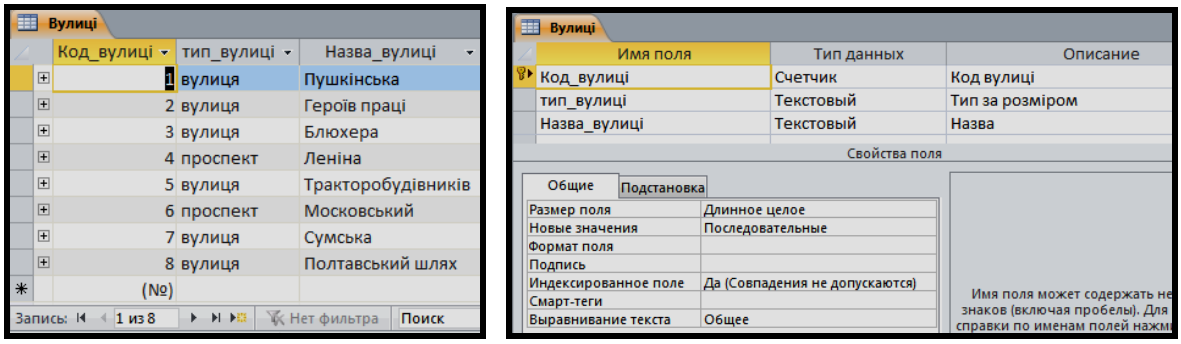
Панелі на вкладках. Містять кнопки для швидкого вибору команд. Перелік кнопок залежить від призначення конкретної панелі.

Рядок стану. Відображає інформацію про поточний режим роботи системи (**Готово, Конструктор** та ін.), підказки, підписи полів та ін.

Панель об'єктів бази. Призначена для роботи з об'єктами MS Access у різних режимах. Панель **Все объекты Access** служить для вибору типів об'єктів за допомогою випадаючих меню.

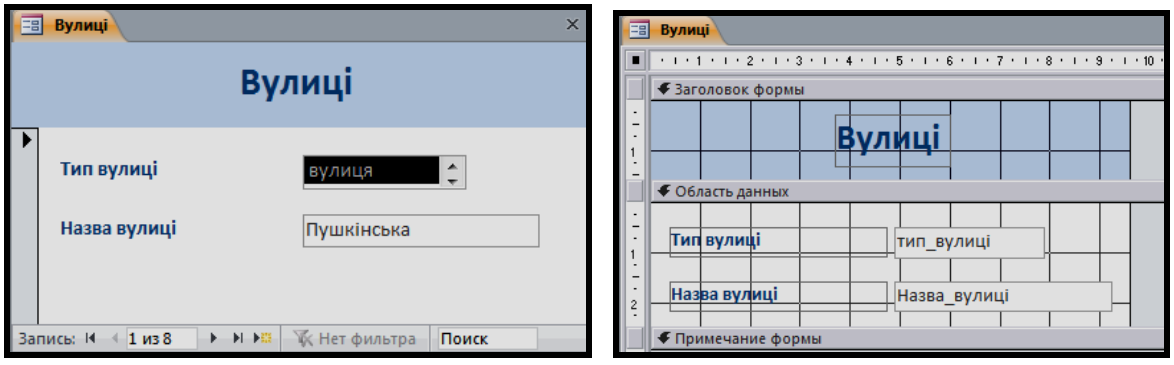
Вікна об'єктів і конструктора

Залежно від режиму роботи з виділеним об'єктом відкривається або вікно об'єкта, або вікно конструктора (рис. 3.2, 3.3).



а) б)

Рис. 3.2 – Вікна таблиці «Вулиці» в режимі таблиці (а) і в режимі конструктора (б)



а) б)

Рис. 3.3 – Вікна форми «Вулиці» в режимі форми (а) і в режимі конструктора (б)

Вікно об'єкта. Містить інформацію про відкритий об'єкт. У вікні об'єкта виконуються операції введення, перегляду або редагування даних. Вікно має стандартну структуру. Рядок заголовка вікна містить піктограму системного меню, ім'я об'єкта і кнопки управління вікном (згортання, розгортання та закриття). У робочій області вікон відображається інформація, а внизу розташовані навігаційні панелі.

Вікно конструктора. Надає можливість створювати новий об′єкт бази даних чи редагувати вже існуючий. Залежно від виду об′єкта, відкритого в режимі конструктора, його зовнішній вигляд може розрізнятися. Основне призначення вікна конструктора – створювати чи редагувати об′єкти бази даних. Незалежно від виду об′єкта перейти у режим конструктора можна за допомогою контекстного меню мишки, вибравши команду **Конструктор**  .

Створення бази даних

У СКБД MS Access використовуються автоматизований спосіб створення БД і спосіб створення БД користувачем самостійно з «нуля». Незалежно від вибраного способу буде створена лише порожня база даних. Введення даних у таблиці найчастіше доводиться виконувати вручну.

Створення бази даних користувачем самостійно

Процес створення бази даних можна розділити на два етапи.

Етап 1. Створення файла нової бази даних.

На вкладинці **Файл** стрічки бази даних виберіть команду **Создать** і виберіть **Новая база данных**.

У полі Имя файла задайте повне ім'я файла БД (враховуючи папку розміщення БД) і клацніть на кнопці Создать.

Після створення і збереження файла відкриється вікно бази даних.

Етап 2. Формування об'єктів нової бази даних.

1. Сформуйте структуру всіх таблиць БД і встановіть зв'язки між ними.
2. Введіть у таблиці контрольні дані для тестування бази даних.
3. Створіть форми.
4. Побудуйте і перевірте виконання запитів.
5. Підготуйте звіти для виведення даних на друк.
6. Розробіть кнопкову форму для роботи з базою даних.
7. Заповніть таблиці і перевірте роботу бази даних.

Створення бази даних в автоматизований спосіб

У MS Access є великий набір шаблонів баз даних, які орієнтовані на рішення типових прикладних завдань. Шаблони використовуються для автоматизованої побудови бази даних за допомогою майстра. На основі вибраного користувачем шаблона майстер створює готову базу даних із таблицями, формами, запитами, звітами та іншими об'єктами.

Процес створення бази даних можна поділити на три етапи.

Етап 1. Вибір шаблона і запуск майстра баз даних.

В області **Доступные шаблоны** вкладинки **Файл** стрічки виберіть команду **Создать**. Потім виберіть потрібний шаблон БД, який найбільше підходить для реалізації вашого проекту з тих, які пропонує MS Access. Натисніть на кнопку **Загрузить**.

Етап 2. Робота з майстром баз даних

На рис. 3.4 показане вікно завантаження шаблона обраної бази даних.

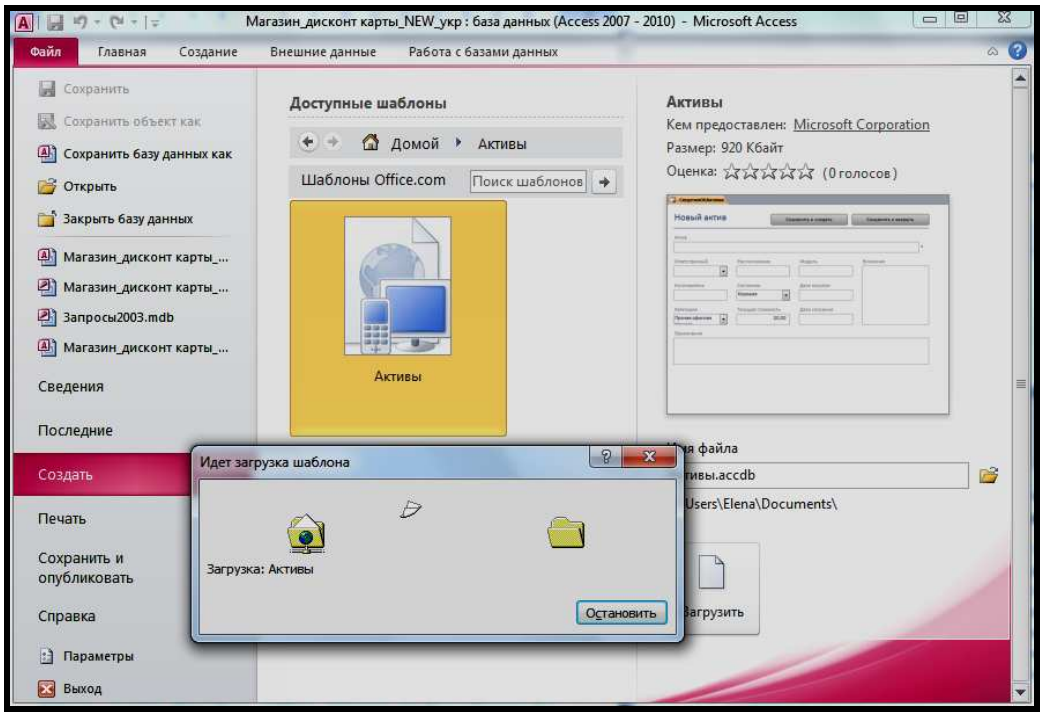


Рис. 3.4 – Завантаження шаблона бази даних «Активи»

Етап 3. Аналіз і доопрацювання створеної бази даних.

Для перегляду об'єктів створеної бази даних на панелі **Все объекты MS Access** виберіть необхідний тип об’єкта та перегляньте створені майстром таблиці, форми, звіти й запити. Відкрийте і проаналізуйте їх структуру.

Таким чином, за допомогою майстра створюється готова база даних, але таблиці в ній порожні. Після уточнення структур таблиць і доопрацю-

вання інших об'єктів бази даних можна приступати до заповнення її таблиць реальними даними.

3.1 Створення таблиць реляційної бази даних

Створення таблиць

Створення таблиці – це формування структури таблиці засобами MS Access і збереження порожньої таблиці у файлі бази даних. Таблиці можна створювати в режимі таблиці і конструктора.

Типи даних

У MS Access допускається використовувати наступні типи даних: **текстовый**, **поле MEMO**, **числовой**, **дата/время**, **денежный**, **счетчик**, **логический**, **поле объекта OLE**, **гиперссылка**, **мастер подстановок**, **вычисляемый**, **вложение**.

Текстовые поля. Містять букви, цифри і спеціальні символи. Ширина поля – від 1 до 255 символів. Властивість *Розмір поля* визначає максимальну кількість знаків, які можна ввести в поле. Приклади змісту текстових полів: АТ «Прометей», Сума, 340-68-75.

Числовые поля. MS Access оперує з цілими і дійсними числами. Довжина числових полів кратна байту: 1, 2, 4 або 8 байтів. Властивостями числового типу є: розрядність (табл. 3.1) і формат поля, розрядність дробової частини числа.

Таблиця 3.1 – Підтипи полів для типу **Числовой**

Розмір поля	Десяткові знаки	Діапазон значень	Розмір пам'яті
Байт	Немає	від 0 до 255	1 байт
Целое	Немає	від –32 768 до 32 767	2 байти
Длинное целое		від –2 147 483 648 до 2 147 483 647	4 байти
Одинарное с плавающей точкой	7	від $-3{,}402823\times10^{38}$ до $+3{,}402823\times10^{38}$	4 байти
Двойное с плавающей точкой	15	від $-1{,}79769313486231\times10^{308}$ до $+1{,}79769313486231\times10^{308}$	8 байтів
Действительное	28	від -10^{28} до 1028	14 байтів

Властивість **Формат поля** встановлює формат представлення чисел: **основной** (3456,789), **денежный** (3456,78 грн), **Евро** (3456,78 €), **фиксированный** (3456,78), **с разделителями разрядов** (3 456,78), **процентный**

(123,00%), **экспоненциальный** (3,46E+03), де E – підстава ступеня 10.

Властивість **Число десятичных знаков** встановлює точність представлення чисел – від 0 до 15 знаків після коми. Помилково введені числові дані системою ігноруються.

Поля денежного типа. Спеціальні числові поля, які використовуються для операцій над грошовими величинами і для запобігання округленню під час обчислень. Для **денежных** полів кількість десяткових знаків після коми автоматично встановлюється рівною 2, наприклад 1234,56 грн, 234,56 руб, 4567,89 € і т. д. При розмірі **денежного** поля у 8 байтів обчислення проводяться з точністю до 15 знаків у цілій частині і до 4 знаків у дробовій частині.

Поля счетчиков. Різновид числових полів з розміром **Длинное целое**. Значення поля **счетчика** автоматично збільшується на 1 при додаванні кожного нового запису в базу даних, наприклад 1, 2, 3 або 100, 101, 102 та ін. Вміст поля **счетчика** може змінюватися за випадковим законом, якщо включити датчик випадкових чисел. Довжина поля – 4 байти.

Поля дата/время. Призначаються для розміщення дат і часу у вибраному форматі. MS Access допускає введення дат і часу тільки у встановлених форматах, наприклад: 10.06.2006, 19:30:06, 15 травня 2013 та ін. Дати і час з помилками ігноруються. Можливі діапазони зміни дат – від 01.01.0100 р. до 31.15.9999 р. Довжина поля – 8 байтів.

Логические поля. Використовуються для представлення даних, що приймають одне із двох значень: **Истина/Ложь, Да/Нет, Вкл/Выкл**.

Поля MEMO. Мають довільну довжину і містять всілякі примітки, описи. Розмір MEMO-поля може доходити до 65 535 символів, тобто практично необмежений.

Поля объектов OLE. Служать для розміщення в базі об'єктів, створених у різних додатках **Windows**, наприклад, таблиць, діаграм, рисунків, текстів і фотографій.

Поля гиперссылок. Служать для зберігання адрес гіперпосилань.

Поля мастера подстановок. Містять списки текстових значень, що вводяться користувачем або взяті з інших таблиць. Використовуються як засоби автоматизації введення даних у поля таблиць.

Тип даних **Вложение** дозволяє берегти всі типи документів і файли в базі даних без зайвого збільшення розміру бази даних. Додаток MS Access автоматично виконує стиснення вкладень, коли це можливо, щоб залишити

якомога більше вільного простору. Використання вкладень значно полегшує такі завдання, наприклад, як зберігання в базі даних цифрових фотографій. Можна навіть добавити до одного запису декілька вкладень. Поля вкладень можна використовувати у веб-базах даних, проте у веб-таблиці може бути не більше одного такого поля.

Вычисляемый. Можно створити поле, в якому буде відображатися значення, що обчислюється на основі інших даних із цієї ж таблиці. Для створення обчислень використовується **Построитель выражений**, який забезпечує зручний доступ до довідкових відомостей про значення у виразах. Дані з інших таблиць не можна використовувати як джерело даних, що обчислюються.

При виборі типу даних необхідно враховувати:

- максимально можливе значення числових даних. Для цілочисельних даних вибирається розмір числового поля – байт, ціле або довге ціле. Тип даних «Одинарне з плаваючою крапкою» або «Подвійне з плаваючою крапкою» вибирають, щоб уникнути переповнень при виконанні арифметичних операцій;
- довжину значень у полі. Наприклад, для текстових полів за замовчуванням встановлюється довжина 50 символів, що не завжди потрібно для зберігання прізвищ, імен, по батькові і т. п.;
- для даних вигляду «Так/Ні», «Має/Не має» слід використовувати тип полів **логический**;
- чи будуть використані поля для зв'язку з полями інших таблиць. Такі поля повинні мати однаковий тип. Виняток: поля типу «лічильник» можна пов'язувати з числовими полями розміру «довге ціле»;
- чи потрібне сортування або індексування значень полів, групування в запитах за значеннями поля. Неможливо сортувати, індексувати і групувати поля **МЕМО**, гіперпосилання та поля об'єктів **OLE**.

Загальні властивості полів таблиць

Властивість **Подпись** задає назву поля у вікнах таблиці, форми і звіті. Якщо властивість не задана, то як підпис використовується ім'я поля.

Властивість **Значение по умолчанию** автоматично додає значення в полі нового запису. Значення можна задати у вигляді константи або виразу.

Властивості **Условие на значение** та **Сообщение об ошибке** використовуються спільно. У першому випадку задаються вимоги до даних, які

вводяться в поле, а в другому вказується текст повідомлення, який виводиться при порушенні цих вимог. Наприклад, у властивості **Условие на значение** поля «Рік народження» задана умова «>1910», а у властивості **Сообщение об ошибке** – «Неприпустиме значення». При введенні даних перевіряється умова. Якщо вона порушується (значення 1900), то виводиться повідомлення «Неприпустиме значення».

Властивість **Обязательное поле** за замовчуванням має значення «Нет» (поле таблиці може бути порожнім). Якщо вибрати в списку значення «Да», то порожні значення в цьому полі не допускаються. Ця властивість не визначена для полів із типом даних **Счетчик**.

Властивість **Индексированное поле** використовується для індексації полів таблиці з метою скорочення часу на пошук і сортування даних. У списку можна вибрати одне із трьох значень властивості:

Нет – поле не індексується;

Да (Совпадения не допускаются) – поле індексується, значення поля в записах таблиці не повинні збігатися.

Да (Допускаются совпадения) – поле індексується, значення поля в записах таблиці можуть збігатися.

Способи створення таблиць

Таблиці можна створювати в режимі конструктора або таблиці, а також за допомогою майстра таблиць.

Вибір способу створення таблиць

Спосіб створення таблиці можна вибрати на панелі **Таблицы** вкладинки **Создание** стрічки MS Access (рис. 3.5).

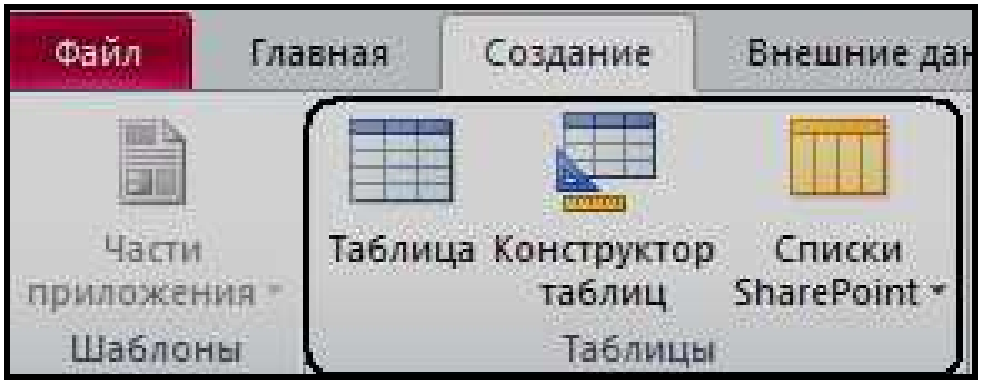


Рис. 3.5 – Вибір способу створення таблиці у вікні бази даних

Структури таблиць створюються залежно від призначення бази даних.

Створення таблиці в режимі конструктора

У режимі конструктора відкривається вікно конструктора (рис. 3.6). Спочатку вікно порожнє і містить у рядку заголовка ім'я нової таблиці **Таблица 1:** і ім'я об'єкта – **таблица**.

Верхня панель вікна містить три стовпці: **Имя поля** (імена полів вводяться вручну)**, Тип данных** (типи даних вибираються зі списку) і **Описание** (коментарі вводяться вручну).

Нижня панель **Свойства поля** містить дві вкладки: **Общие** і **Подстановка**. Вкладка **Общие** використовується для задавання властивостей полів таблиці. Значення властивостей вибираються у списках. Вкладка **Подстановка** дозволяє задати параметри підстановки даних у таблицю.

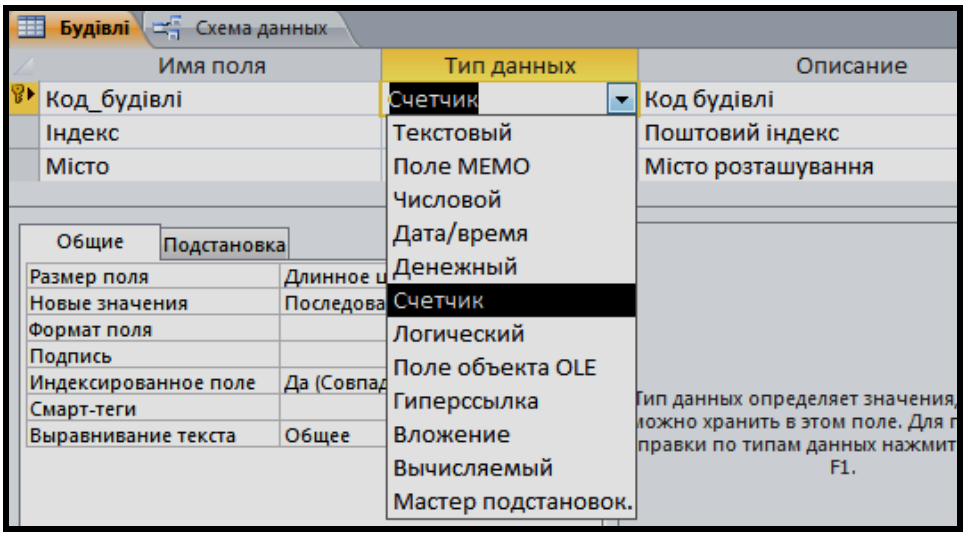


Рис. 3.6 – Вікно конструктора при створенні структури таблиці

Розглянемо порядок створення таблиці на прикладі таблиці **Будівлі** бази даних **«Кадастр»**.

1. Ведіть ім'я поля Код_Будівлі в першому рядку стовпця **Имя поля**.

2. Виберіть тип даних. Перемістіть курсор у стовпець **Тип данных** мишею (або клавішами Tab чи Enter). В осередку стовпця з'явиться значення **Текстовый**, змініть його на **Счетчик.**

3. Введіть опис поля у стовпці **Описание.** Опис має бути коротким, зрозумілим та інформативним. Текст опису відображатиметься в рядку стану при установці курсору в полі відкритої таблиці.

4. Задайте властивості поля на панелі Свойства поля за допомогою вкладки **Общие.**

5. Виконайте кроки 1 – 4 для решти полів таблиці.

6. Створіть ключове поле таблиці:

- виділіть поле простого первинного ключа або всі поля складеного первинного ключа, утримуючи натиснутою клавішу **Ctrl**;
- виберіть команду **Ключевое поле** з контекстного меню або натисніть кнопку **Ключевое поле**  .

7. Збережіть таблицю. Закрийте вікно конструктора і введіть ім'я таблиці у вікні Сохранение або виконайте команду Файл/Сохранить как і введіть ім'я таблиці у вікні Сохранение. Після натиснення кнопки **ОК** таблиця буде збережена у файлі бази даних.

Примітка. Якщо первинний ключ не заданий, MS Access перед збереженням таблиці запропонує створити ключ. При натисненні кнопки **Да** система створить ключове поле з ім'ям **Код** типу **Счетчик**.

Створення таблиці в режимі таблиці

Це простий і наочний спосіб створення таблиць.

1. За допомогою кнопки **Таблица** панелі **Таблицы** вкладинки **Создание** (див. рис. 3.1) відкриється вікно **Таблица 1**. Це макет стандартної таблиці (рис. 3.7).
2. Виберіть тип даних, клацнувши по трикутничку поля **Щелкните для добавления**. Введіть імена полів або використайте команду контекстного меню **Переименовать столбец**.
3. Для додавання нових полів повторіть пункт 2.
4. Збережіть таблицю. Закрийте вікно таблиці, введіть її ім'я у вікні **Сохранение** і натисніть кнопку ***ОК***.
5. Перейдіть у режим конструктора й уточніть типи полів (якщо MS Access визначив їх неправильно), задайте властивості полів і первинний ключ таблиці.

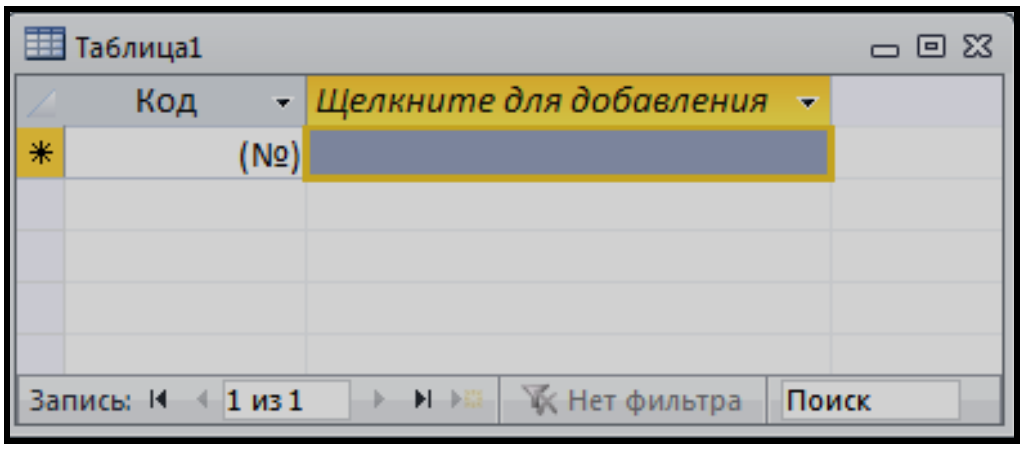


Рис. 3.7 – Створення таблиці в режимі таблиці

Створення таблиці в режимі SQL

Створювати таблиці баз даних можна не тільки через графічний інтерфейс, а й за допомогою команд SQL. Якщо ж необхідно створити базу даних, що складається з великої кількості таблиць, які мають складні взаємозв'язки, то краще для цих цілей використовувати спеціалізовані CASE-засоби, які дозволяють в інтерактивному режимі згенерувати всю структуру бази даних і сформувати всі зв′язки.

Створення таблиць за допомогою операторів мови DML розглянуто у підрозділі «Створення запитів у режимі SQL Access 2010».

Індексація полів таблиці

Питанню індексації полів треба приділити особливу увагу, оскільки застосування індексів полів дозволяє істотно скоротити час пошуку даних в таблицях (особливо це важливо для великих баз даних).

Прості і складені індекси полів

Для індексації полів використовуються прості і складені індекси.

Простий індекс створюється для одного поля таблиці (рис. 3.8), якщо його властивості **Индексированное поле** задане значення **Да (Допускаются совпадения)** або **Нет (Совпадения не допускаются)**.

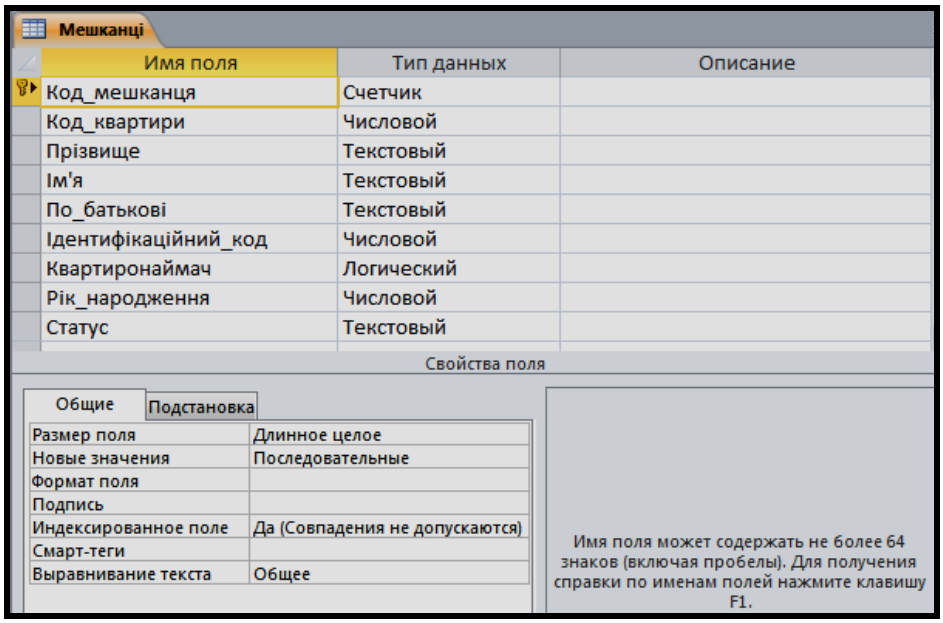


Рис. 3.8 – Вікно **Конструктор** таблиці «Мешканці»

Складений індекс застосовується для індексації декількох полів таблиці. При сортуванні таблиці за складеним індексом спочатку виконується сортування за першим полем індексу (наприклад, за полем *Прізвище*). Якщо

в першому полі містяться записи із значеннями, що повторюються, то виконується сортування за другим полем індексу (наприклад, за полем *Ім'я*) і т. д. Складеним індексом можна індексувати до десяти полів.

Приклад. Як приклад розглянемо вікно **Индексы**, в якому створений складений індекс для полів **Прізвище**, **Ім'я** та **По батькові**. Для створення складених індексів натисніть кнопку **Индексы**  . Вікно складається із трьох областей (рис. 3.9), які використовуються для створення індексу.

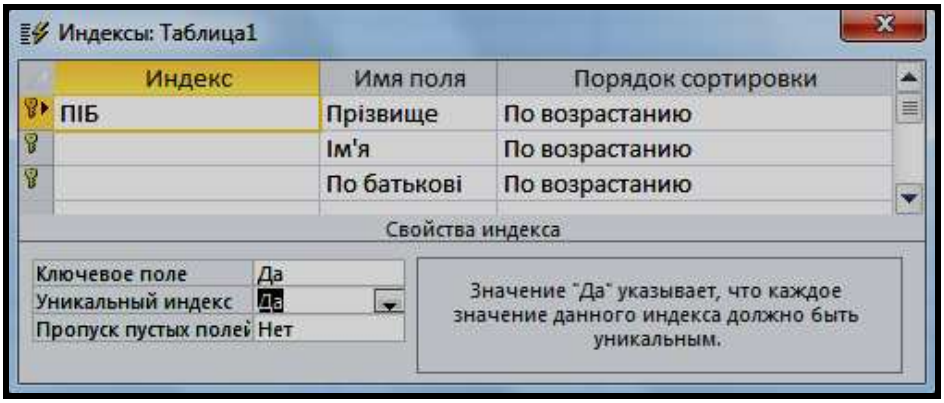


Рис. 3.9 – Вікно складеного первинного ключа

У лівій області введіть ім'я індексу, наприклад ПІБ, замість PrimaryKey, який MS Access створює автоматично. У середній області виберіть у списках імена полів, для яких створюється складений індекс (наприклад, Прізвище, Ім'я, По батькові). У правій області виберіть у списках напрям сортування за кожним полем: «За збільшенням» або «За убуванням». Аналогічно створюються і інші складені індекси.

Примітка. Індексовані поля повинні мати тип **Текстовый**, **Числовой**, **Денежный** або **Дата/время**.

Особливості індексації ключових полів

При створенні **первинних ключів** (простих або складених) автоматично формуються їх індекси, які можна побачити у вікні *Індекси*.

Для **простого первинного ключа** у властивості **Індексоване поле** автоматично встановлюється значення **Да** (**Совпадения допускаются**), оскільки кожне значення цього ключового поля має бути унікальним.

У **складеному первинному ключі** підлеглої таблиці у властивості **Індексоване поле** кожного поля треба вручну встановити значення **Да** (**Совпадения не допускаются**). У цьому випадку при зв′язку первинного ключа головної таблиці з одним із полів складеного первинного ключа пі-

длеглої таблиці буде встановлений зв'язок **«один-ко-многим»**.

Модифікація таблиць

Загальні рекомендації

Після створення і тестування бази даних може виникнути потреба змінити структури деяких таблиць: змінити імена, типи або властивості полів; додати, видалити, перемістити або перейменувати поля.

Якщо таблиці порожні і на їх основі ще не створені інші об'єкти (форми, запити і звіти), то реорганізація таблиць не викликає особливих труднощів і ускладнень. Якщо таблиці заповнені, то некоректне редагування (особливо зміна типів даних або видалення полів) може призвести до безповоротної втрати даних. Тому перед внесенням структурних змін створіть копії таблиць, щоб у разі потреби можна було повернутися до їх первинного вигляду.

Модифікація таблиць та інших об'єктів бази даних виконується в режимі конструктора. Відкрийте таблицю в режимі конструктора. У вікні конструктора відредагуйте структуру таблиці і збережіть зміни.

Зміна імен полів

Для зміни імені поля двічі клацніть на імені поля і введіть нове ім'я або встановіть курсор у полі імені й відредагуйте старе ім'я.

Додавання, видалення і зміна порядку полів

Щоб включити в таблицю нове поле, досить додати новий рядок у список полів. Порядок дій такий:

1. Виділіть поле, над яким передбачається помістити нове поле таблиці. Для цього перейдіть у **Конструктор таблиц** та клацніть на кнопці вибору поля (рис. 3.10).

Рис. 3.10 – Вікно конструктора таблиці

2. Виконайте команду **Вставить Строки** на вкладинці **Конструк-**

тор панелі **Сервис** або команду контекстного меню **Вставить Строки**. Новий рядок з′являється над поточним рядком.

3. У порожньому рядку введіть ім'я нового поля, виберіть тип даних і задайте значення властивостей поля.

Видалення поля виконується простіше. Виділіть поле і натисніть клавішу **Delete** або виконаєте команду **Удалить строки** на вкладинці **Конструктор** панелі **Сервис**. Аналогічно можна видалити і декілька полів. Для виділення суміжних полів виділіть перше з них, а потім з натиснутою клавішею **Shift** – останнє поле. Несуміжні поля виділяються при натиснутій клавіші **Ctrl**.

Щоб перемістити поле, наведіть покажчик миші на кнопку вибору поля, з'явиться стрілка вибору поля. Утримуючи натиснутою ліву кнопку миші, перетягніть поле на нове місце у списку полів.

Зміна типів і розмірів полів

Для зміни типу поля відкрийте список типів даних цього поля і виберіть у ньому новий тип, а в області властивостей задайте новий розмір поля. Щоб уникнути спотворень або втрати даних, дотримуйтеся таких загальних правил:

- виконуйте тільки допустимі перетворення;
- довжина поля нового типу має бути більше довжини поля старого типу. Інакше можливе усікання даних.

Створення зв'язків між таблицями схеми бази даних

Основні поняття

Схема даних – це структура бази даних, елементами якої є таблиці і зв'язки між ними. Скріплення кожної пари таблиць бази даних здійснюється через поля зв'язку. Одна із пов'язаних таблиць є головною, а друга – підпорядкованою. У головній таблиці поле зв'язку є первинним ключем, а в підпорядкованій – або ключовим полем складеного первинного ключа, або неключовим полем (зовнішнім ключем). Поля зв'язку в первинній і підлеглій таблицях можуть мати різні імена, але типи даних та значення характеристик (особливо розміру) повинні в них збігатися. Крім того, поля зв'язку мають бути індексованими, щоб скоротити час на пошук даних.

Створення схеми даних

Створення схеми даних виконується у вікні **Схема данных**. Щоб ві-

дкрити вікно, натисніть кнопку  на панелі **Отношения** вкладинки **Работа с базами данных**. Разом із порожнім вікном **Схема данных** відкривається і діалогове вікно **Добавление таблицы** (рис. 3.11).

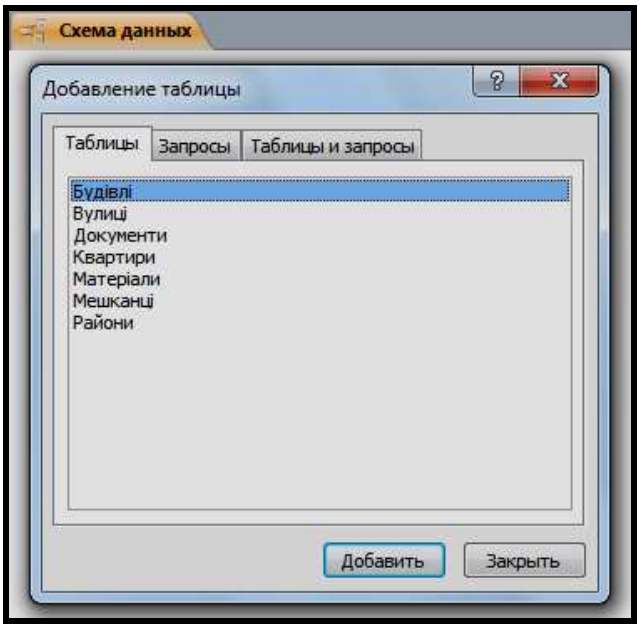


Рис. 3.11 – Діалогові вікна **Схема данных** та **Добавление таблицы**

Схема даних будується в три етапи.

Етап 1. Розміщення таблиць у вікні **Схема даних.**

У діалоговому вікні **Добавление таблицы** відкрийте вкладку **Таблицы,** в якій знаходиться список таблиць бази даних. Для додавання у вікно **Схема данных** таблиці виділіть її в списку таблиць і натисніть кнопку **Добавить**. Після додавання всіх таблиць натисніть кнопку **Закрыть** у вікні **Добавление таблицы**.

У вікні **Схема данных** таблиці розміщуються в порядку їх додавання. За допомогою миші можна розмістити їх так, щоб наочно відображалася підлеглість таблиць і зв'язку (див. рис. 2.18). Це дозволить правильно вибрати порядок заповнення таблиць даними.

Етап 2. Установка зв'язків між таблицями.

Для зв’язування таблиць перенесіть за допомогою миші поле зв'язку однієї таблиці у відповідне поле іншої таблиці. Наприклад, поле *Код_вулиці* таблиці *Вулиці* перенесіть у поле *Код_ вулиці* таблиці *Будівлі*. Відкривається вікно **Изменение связей** для визначення параметрів зв'язку (рис. 3.12).

За вибраними полями автоматично визначається тип зв'язку між таб-

лицями. Якщо поля зв'язку головної і підлеглої таблиць є простими первинними ключами, то визначається зв'язок типу «**один-к-одному**». Якщо ж поле зв'язку в головній таблиці є простим первинним ключем, а в підлеглій таблиці використовується ключове поле складеного первинного ключа або зовнішній ключ, то визначається зв'язок «**один-ко-многим**». На рис. 3.12 показано, що між полями *Код_вулиці* таблиць *Вулиці* і *Будівлі* автоматично визначений зв'язок типу «**один-ко-многим**». Тобто одному запису головної таблиці *Вулиці* відповідають декілька записів підлеглої таблиці *Будівлі*.

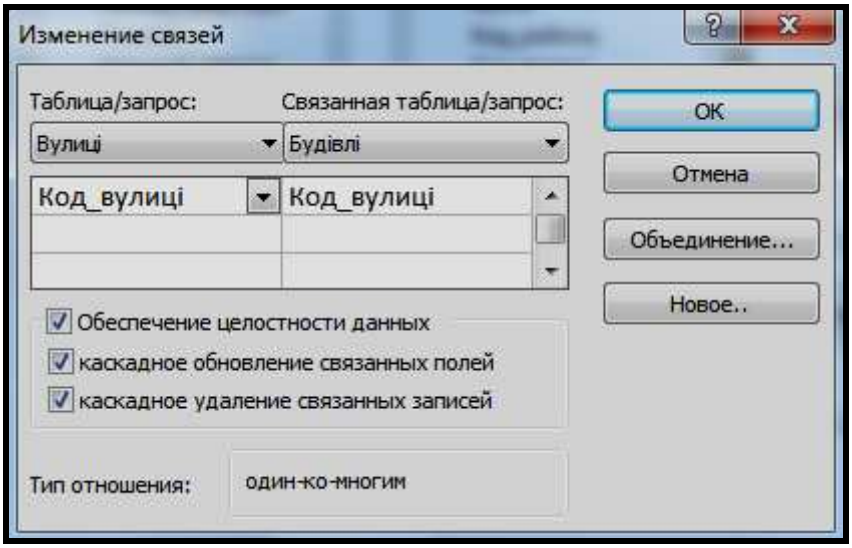


Рис. 3.12 – Вікно **Изменение связей**

Для забезпечення цілісності даних необхідно встановити прапорець опції **Обеспечение целостности данных**. У цьому випадку автоматично перевіряється цілісність зв'язку між обома таблицями і виключаються помилки, що виникають при видаленні записів із головної таблиці і введенні інформації в підлеглу таблицю.

Етап 3. Забезпечення цілісності даних.

Щоб подолати обмеження на видалення або зміну пов'язаних записів, зберігаючи при цьому цілісність даних, слід встановити прапорці **Каскадное обновление связанных полей** і **Каскадное удаление связанных записей** (див. рис. 3.12). Якщо встановлений прапорець **Каскадное обновление связанных полей**, то при зміні первинного ключа головної таблиці автоматично змінюються і відповідні значення в полі зв'язку записів підлеглої таблиці. Якщо встановлений прапорець **Каскадное удаление связанных записей**, то при видаленні запису в головній таблиці видаляються і всі пов'я-

зані записи в підлеглій таблиці. Наприклад, при видаленні даних про конкретну вулицю в головній таблиці *Вулиці* в підлеглій таблиці *Будівлі* автоматично будуть видалені всі дані, що відносяться до цієї вулиці. Завершується установка зв'язку натисненням кнопки **ОК** (див. рис. 3.12).

Примітка. Якщо в головній таблиці вибрано неключове поле зв'язку, MS Access не може визначити тип відношень. У цьому випадку треба встановити тип відношення самостійно. Натисніть кнопку **Объединение** (див. рис. 3.12) і в діалоговому вікні **Параметры объединения** виберіть один із трьох запропонованих способів зв'язку.

Введення даних до таблиці і їх редагування

Використовуються такі способи введення даних у таблиці:

- введення даних із клавіатури безпосередньо у вікні таблиці або у вікні форми, створеної на основі таблиці;
- використання списків із фіксованими наборами даних, створених за допомогою майстра підстановок;
- введення даних із інших таблиць за допомогою майстра підстановок.

При виборі порядку заповнення таблиць необхідно враховувати підлеглість таблиць, встановлену в схемі даних. З метою забезпечення цілісності даних спочатку потрібно заповнити незалежні головні таблиці, потім – підлеглі таблиці другого, третього та інших нижніх рівнів ієрархії.

Введення даних із клавіатури безпосередньо у вікні таблиці

Введення даних із клавіатури виконується безпосередньо у відкритому вікні таблиці. На рис. 3.13 показано два вікна заповнених таблиць бази даних: вікно головної таблиці *Вулиці* і вікно підлеглої таблиці *Будівлі*.

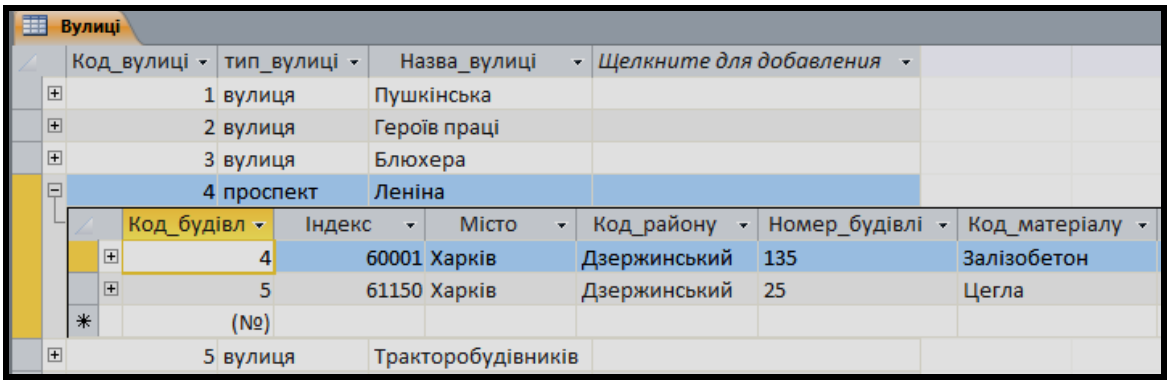


Рис. 3.13 – Вікна головної таблиці *Вулиці* і підлеглої таблиці *Будівлі*

У головній таблиці *Вулиці* зліва від записів видно значок «+». Після

клацання на значку «+» відображається інформація із зв'язаної таблиці *Будівлі* і значок «+» замінюється на «–». На рис. 3.13 показана інформація про будинки, розташовані на проспекті Леніна. Клацання на значку «–» переводить таблицю в початковий стан.

Таблиці слід заповнювати по рядках зліва направо, починаючи з введення первинного ключа. Виключенням є ключове поле типу **Счетчик**. У це поле значення вводяться автоматично, наприклад, 1, 2, 3 і т. д. Для переходу між полями використовуйте клавішу **Tab**, мишу або клавіші переміщення курсора.

Формування списків із фіксованими наборами даних

Списки із фіксованими наборами даних доцільно використовувати в тих випадках, коли заздалегідь відомі всі можливі значення, які передбачається вводити в поле таблиці. Для формування списку використовуються три кроки майстра підстановок. Робота виконується в діалогових вікнах **Создание подстановок**.

Принцип роботи з майстром підстановок розглянемо на прикладі створення списку видів вулиць для введення даних у полі *Тип_ вулиці* таблиці *Вулиці* бази даних **«Кадастр»**.

Крок 1. Для запуску майстра підстановок перейдіть у режим конструктора таблиці і виберіть поле **Мастер подстановок** у списку типів даних поля, для якого створюється список значень. Майстер відкриває перше вікно створення підстановки, в якому пропонується вибрати спосіб створення стовпця підстановки – або фіксований набір значень, або вибір списку з таблиці чи запиту. У даному випадку виберіть спосіб створення стовпця підстановки як фіксованого набору значень та перейдіть до другого кроку, натиснувши кнопку **Далі**.

Крок 2. Створення списку значень поля. На цьому кроці задайте число стовпців списку і встановіть ширину стовпця підстановки (рис. 3.14).

У клавіатури введіть у рядки стовпця підстановки всі можливі значення даного поля, в даному прикладі введіть список видів вулиць. Для переходу до чергового рядка стовпця підстановки використовуйте клавішу **Tab** або клавіші переходу на клавіатурі. Після створення списку перейдіть до третього кроку роботи майстра.

Крок 3. На завершальному кроці роботи майстра задайте ім'я поля для створеного списку підстановки або виберіть ім'я, запропоноване програмою, наприклад *Тип_ вулиц*, і натисніть кнопку **Готово**.

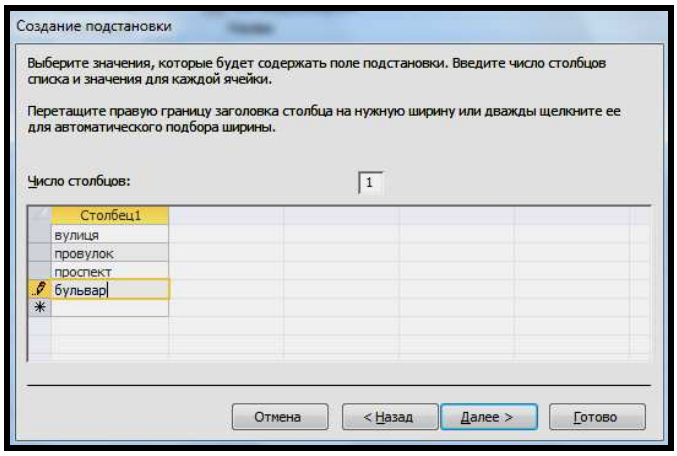


Рис. 3.14 – Діалогове вікно **Создание подстановки** (крок 2)

Тепер для введення типу вулиці в полі *Тип_вулиці* таблиці *Вулиці* досить вибрати відповідне значення з випадаючого списку.

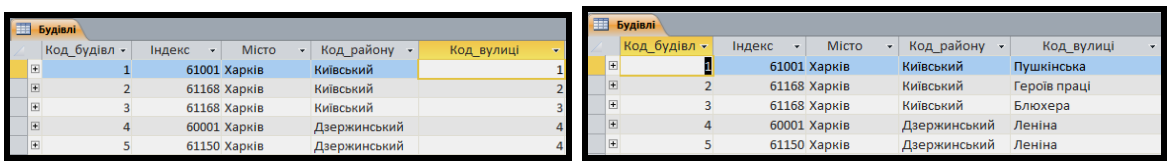
Введення даних з інших таблиць БД

Дані з інших таблиць доцільно використовувати при введенні значень первинних ключів головних таблиць у відповідні поля зв'язку підлеглих таблиць. Принцип введення даних з інших таблиць пояснимо на прикладі заповнення полів зв'язку підлеглої таблиці *Будівлі*.

Поле зв'язку *Код_Вулиці* таблиці *Будівлі* можна заповнювати двома способами.

Перший спосіб. Вказані поля заповнюються кодами вручну (рис. 3.15а). У цьому випадку треба пам'ятати коди товарів, які введені в головній таблиці *Вулиці*. Це робить процес заповнення таблиці незручним, а також передбачає подальший її аналіз чи редагування.

Другий спосіб заснований на підстановці коду вулиці в поле зв'язку таблиці *Будівлі* з ключового поля головної таблиці *Вулиці*. Для полегшення підстановки коду використовуються списки вулиць, створені майстром підстановки в полях зв'язку (рис. 3.15б).



Будівлі

Код_будівл	Індекс	Місто	Код_району	Код_вулиці
1	61001	Харків	Київський	1
2	61168	Харків	Київський	2
3	61168	Харків	Київський	3
4	60001	Харків	Дзержинський	4
5	61150	Харків	Дзержинський	4

а)

Будівлі

Код_будівл	Індекс	Місто	Код_району	Код_вулиці
1	61001	Харків	Київський	Пушкінська
2	61168	Харків	Київський	Героїв праці
3	61168	Харків	Київський	Блюхера
4	60001	Харків	Дзержинський	Леніна
5	61150	Харків	Дзержинський	Леніна

б)

Рис. 3.15 – Таблиця *Будівлі* з кодами (а) та іменами полів (б)

При виборі в списку необхідного товару в дане поле вводиться код, а у вікні таблиці відображається відповідна йому назва. Завдяки цьому поле-

гшується процес заповнення, аналізу і редагування таблиці *Будівлі*.

Принцип створення списку підстановки розглянемо на прикладі організації введення коду вулиць із таблиці *Вулиці* в поле *Код_вулиці* таблиці *Будівлі* бази даних **«Кадастр»**. У даному випадку майстер створює списки підстановки автоматично за шість кроків. Робота виконується в діалоговому вікні **Створення підстановки**.

Крок 1. Видаліть зв'язок між головною і підлеглою таблицями. Запустіть майстер точно так, як і при формуванні списку значень, перейшовши у режим **Конструктора** для таблиці *Будівлі*. У вікні **Создание подстановки** виберіть перший спосіб – **Объект «поле подстановки» получит значение из другой таблицы или другого запроса**.

Крок 2. У вікні другого кроку виберіть таблицю (у даному прикладі **Вулиці**) із значеннями, які міститиме стовпець підстановки.

Крок 3. У вікні третього кроку (рис. 3.16) виберіть поля, значення яких слід включити у стовпець підстановки. У даному прикладі вибрані поля *Код_вулиці* та *Назва_товару* таблиці *Вулиці*. Значення поля *Код_вулиці* підставлятиметься в поле зв'язку таблиці *Будівлі*, а відповідні назви відображатимуться у списку поля зв'язку.

Крок 4. Задавання при необхідності сортування в межах створюваного списку.

Крок 5. Установіть ширину стовпця підстановки вручну або двічі клацніть мишею по правій границі стовпця для автоматичного підбору ширини. Крім того, необхідно встановити прапорець для приховування ключового стовпця, значення якого включатимуться в поле зв'язку підлеглої таблиці (див. рис. 3.16). У списку значень, які підставлені, відображатимуться тільки назви вулиць, а самі значення первинного ключа будуть приховані (саме вони і підставлятимуться в поле зв'язку підлеглої таблиці).

Крок 6. Задайте ім'я створеного списку підстановки або виберіть ім'я, запропоноване програмою, наприклад *Код_вулиці*, і натисніть кнопку **Готово** (див. рис. 3.16). Майстер відновить видалений раніше зв'язок між таблицями. Для забезпечення цілісності даних необхідно відкрити вікно **Схема данных** і встановити прапорці опцій **Обеспечение целостности данных**, **Каскадное обновление связанных полей** та **Каскадное удаление связанных записей**.

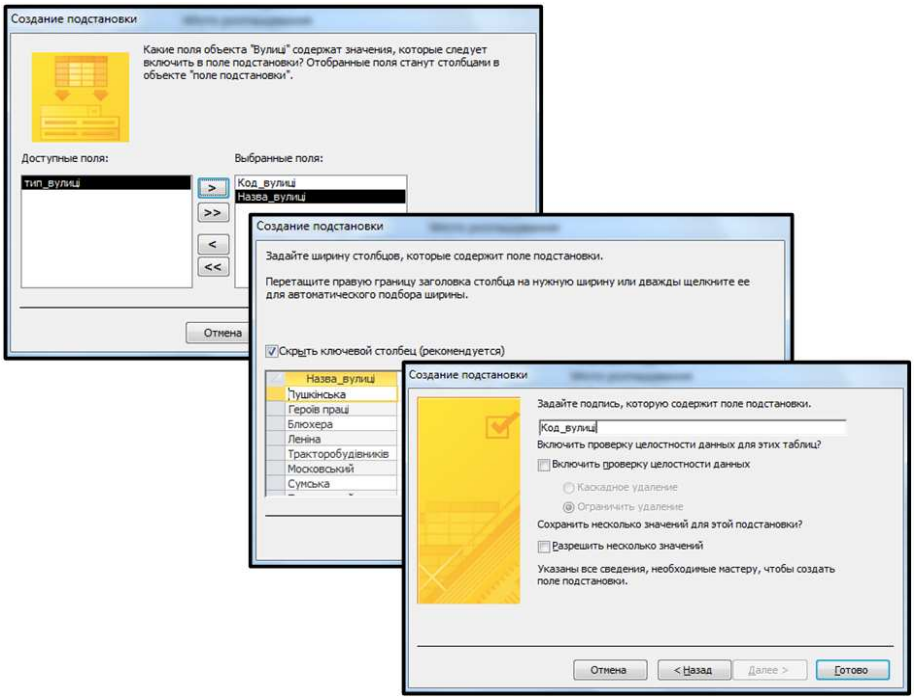


Рис. 3.16 – Діалогові вікна майстра створення підстановки (кроки 3, 5, 6)

Для введення значень первинних ключів головної таблиці в поля зв'язку підлеглої таблиці досить вибрати відповідну назву вулиці з випадаючого списку.

Введення даних про об'єкти поза БД

Перш ніж вводити дані про об′єкти поза базою даних (фотографії співробітників, товару та інше) необхідно перейти в режим конструктора таблиці і встановити для необхідного поля тип даних **Поле обьекта OLE**.

Для введення даних у комірку таблиці відкрийте вікно таблиці, викличте контекстне меню і виконайте команду **Вставить объект**. На екрані відобразиться діалогове вікно (рис. 3.17) для пошуку об'єкта.

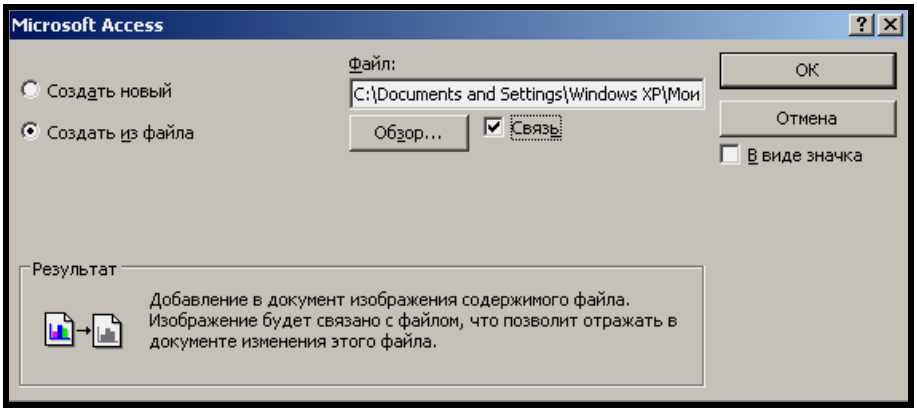


Рис. 3.17 – Діалогове вікно вставки об'єкта

Щоб вставити об'єкт із файла, виберіть перемикач **Создать из файла**, натисніть кнопку **Обзор** і в діалоговому вікні, що відкрилося, знайдіть файл рисунку або іншого об'єкта та натисніть кнопку **ОК.**

У полі таблиці буде записано посилання на об'єкт. Якщо у вікні зробити активним перемикач **Связь**, то поле таблиці з типом **OLE** буде пов'язано з вказаним файлом. Усі зміни у файлі автоматично відображатимуться при відкритті об'єкта.

Перевірка даних при введенні в таблицю

Для перевірки значень, що вводяться, використовуйте властивості полів **Условие на значение, Обязательное поле** та **Маска ввода**.

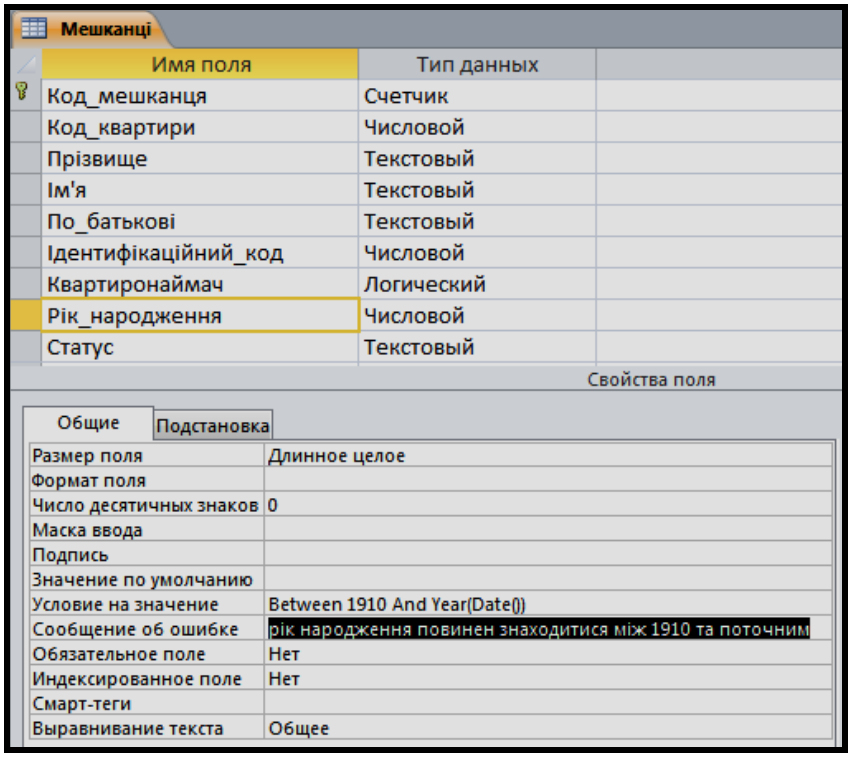


Рис. 3.18 – Завдання умови на значення поля *Рік_народження*

Властивість **Условие на значение** застосовується для контролю правильності введення значень у дане поле. Перевірка виконується при переході в інше поле. Щоб вказати вимоги, які повинні задовольняти значення поля, виконайте такі дії (рис. 3.18):

- відкрийте таблицю в режимі конструктора і встановіть курсор у поле, значення якого контролюється, наприклад у поле *Рік_народження*;
- у властивості **Условие на значение** вкажіть необхідне обмеження. Наприклад, якщо рік народження мешканця знаходиться у діапазоні: «З 1910 по поточний рік», то у властивості **Условие на значение** поля

Рік_народження вкажіть умову *Between 1910 And Year(Date())*;

- у властивості **Сообщение об ошибке** введіть пояснювальний текст попереджувального повідомлення.

Якщо в полі **Рік_народження** буде введено помилковий рік народження, що не входить у задане обмеження, то при виході з поля буде видано попередження з текстом із властивості **Сообщение об ошибке**.

Якщо потрібно перевіряти відразу декілька умов, вкажіть їх у рядку **Условие на значение**, з'єднавши логічними операторами **And** (операція **и**) або **Or** (операція **или**). Можна використовувати **Построитель выражений**, натиснувши кнопку його виклику ... (рис. 3.19).

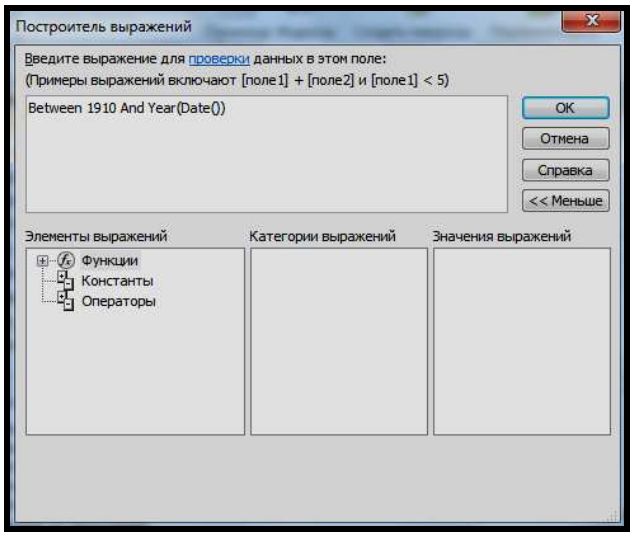


Рис. 3.19 – Вікно Построитель выражений

Крім того, можна зобов′язати користувача обов'язково заповнювати деякі поля. Для таких полів треба встановити властивість **Обязательное поле** рівним значенню **Да.**

Якщо ж передбачається, що значення буде часто повторюватися при заповненні поля, то можна задати властивість **Значение по умолчанию**. У цьому випадку при створенні нового запису поле вже буде заздалегідь заповнено вказаним значенням. При необхідності ці значення можуть бути замінені. Наприклад, можна вказати для поля **Назва_матеріалу** таблиці **Матеріали** значення за замовчуванням, рівне «*Залізобетон*» (передбачається, що таке значення буде найчастішим).

Питання для самодіагностики

1. Що таке об'єкт типу «Таблиця», як його можна створити?

2. Які властивості необхідно визначати при описі полів даних таблиці?

3. Що таке «Ключ» для об'єкта типу «Таблиця» і як його можна задати, якщо він складається з одного або декількох полів таблиці? Чи може бути задано для таблиці декілька ключів?

4. Що таке зв'язки між таблицями, якого типу зв'язки між таблицями підтримує MS Access?

5. Що таке первинний ключ таблиці? Чому для кожної таблиці слід встановлювати первинний ключ?

6. Як можна задати зв'язки між таблицями і де ці зв'язки будуть зафіксовані? Які зміни можна внести до зв'язків між таблицями?

7. Що таке головна таблиця і підлегла таблиця, як вони пов'язані між собою?

8. Що таке підтримка каскадного видалення і каскадного оновлення під час опису зв'язків?

9. Для чого служать характеристики полів **Значение по умолчанию** та **Условие на значение**?

10. Як встановити зв'язок між таблицями?

11. Як встановити цілісність даних між таблицями? Що розуміється під цілісністю даних і для чого вона служить?

12. Як створити поле зі списком за допомогою майстра підстановок?

3.2 Створення запитів реляційної бази даних

Призначення і види запитів

Запити забезпечують швидкий і ефективний доступ до даних, що зберігаються в таблицях. Вони використовуються користувачем для отримання з бази даних інформації, яка цікавить його в даний момент, а також для автоматичної зміни вмісту таблиць БД.

Завдяки запитам можна виконати сортування або обчислити вирази. Крім того, можна звести разом дані з декількох пов'язаних таблиць. Деякі запити дозволяють внести зміни до початкових таблиць. На підставі запитів можна розробляти форми і звіти.

Дані, що відібрані в результаті виконання запиту, відображаються у вигляді таблиці. Вони називаються динамічним, або тимчасовим, набором даних. У цій таблиці відображаються вибрані з основної таблиці чи таблиць записи, які задовольняють критерії запиту. Як тільки запит буде за-

критий, динамічний набір даних ліквідується, хоча дані, які користувач бачив у ньому, залишаться в початковій таблиці. Таким чином, при кожному виконанні запиту він будується на основі «свіжих» табличних даних.

Примітка. Дані завжди зберігаються в таблицях. У запиті MS Access зберігає тільки інструкції про те, як мають бути організовані дані в результаті виконання запиту.

У Microsoft Access існує декілька видів запитів:

- **запити на вибірку** – найбільш часто використовуваний тип запитів. При їх виконанні дані, що задовольняють умови відбору, вибираються з однієї або декількох таблиць і виводяться в певному порядку;
- **модифікуючі запити**, що дозволяють змінювати дані в таблицях (видаляти, оновлювати і додавати записи), а також створювати нові таблиці на основі даних однієї або декількох існуючих таблиць;
- **перехресні запити** – результати статистичної обробки даних, що виводяться у вигляді таблиці, дуже схожої на зведену таблицю Excel.

Крім того, запити можуть включати **параметри** (і тоді називатимуться параметричними запитами), **запити з умовою** та **підсумкові запити**.

Запит можна створювати самостійно, з використанням конструктора або ж скористатися майстром запитів. При самостійній розробці нового запиту необхідно в режимі конструктора вибрати таблиці або інші запити, що містять потрібні дані, і заповнити бланк запиту потрібними полями, параметрами та умовами.

Запити деяких типів можуть бути створені за допомогою майстрів MS Access. Майстри запитів прискорюють процес створення запиту, автоматично виконуючи всі основні операції. Викликаний майстер запитів запрошує відомості та створює запит на основі відповідей користувача. Потім можна перейти в режим конструктора і допрацювати запит.

Створення запитів

Створення запитів у режимі конструктора

Конструктор запитів – основний засіб створення і зміни запитів. Навіть створені за допомогою різних майстрів запити часто вимагають доопрацювання, що можна зробити лише в режимі конструктора.

Наприклад, необхідно організувати виведення списку з даними про будівлі. Розглянемо процес створення запиту в цьому режимі на прикладі бази даних **«Кадастр»**.

Щоб створити запит, треба на вкладці **Создание** стрічки бази даних

вибрати панель **Запросы** та команду  **Конструктор запросов**.

За цією командою буде відкрито вікно конструктора. Над цим вікном можна побачити вікно вибору таблиць і запитів, на яких буде заснований створюваний запит. Для даного завдання слід вибрати таблиці *Будівлі*, *Райони*, *Вулиці* та *Матеріали* і закрити вікно додавання таблиць.

Вікно конструктора розділене по вертикалі на дві частини (рис. 3.20). Верхня частина вікна призначена для відображення таблиць, що беруть участь у запиті. У нижній частині вікна знаходиться бланк запиту – таблиця, осередки якої використовуються для визначення полів запиту.

Додамо в бланк запиту поля з таблиць. Для цього треба з верхньої частини вікна перетягнути по черзі необхідні поля таблиць у стовпці бланка. Замість перетягування мишею можна також скористатися подвійним клацанням миші на назві поля в таблиці – поле буде додано в бланк (див. рис. 3.20).

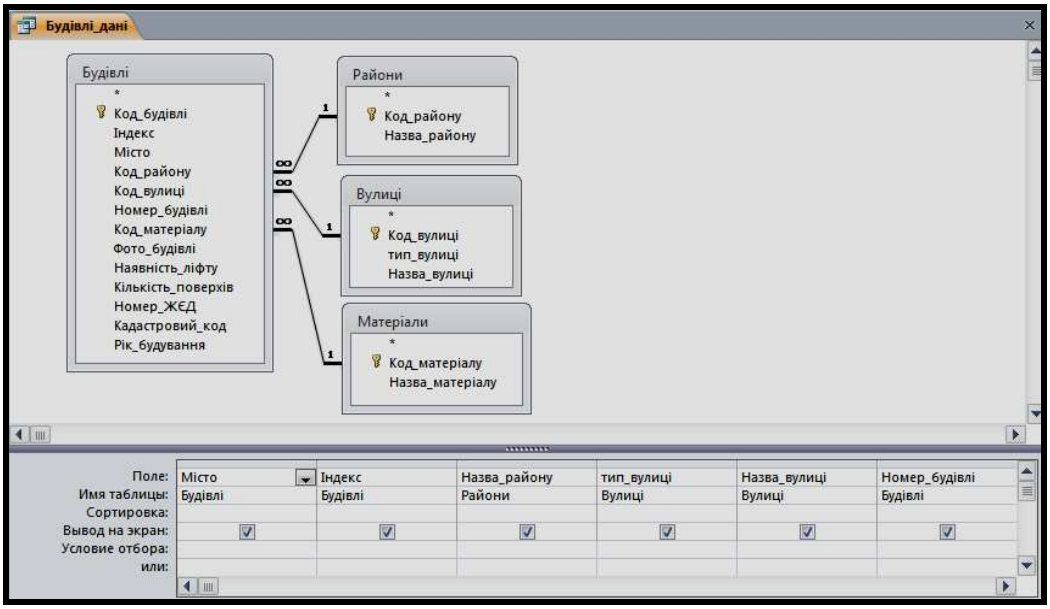


Рис. 3.20 – Вікно конструктора запитів

У бланку для кожного поля вказується його назва (рядок бланка **Поле**) і назва таблиці, що містить його (рядок **Имя таблицы**). Можна вибирати ці назви з випадаючих списків, наявних у кожному полі.

Для впорядкування результатів запиту за яким-небудь полем служить рядок бланка **Сортировка** і випадаючий список у ній: **по возрастанию**, **по убыванию**, **отсутствует**.

Для того щоб провести сортування за декількома полями, поля в бланк-

ку слід розмістити в порядку виконання сортування. MS Access починає сортування з поля, яке розташоване зліва в рядку **Сортировка**, потім переходить до наступного, розташованого праворуч від нього. Наприклад, щоб виконати сортування спочатку за полем *Дата*, а потім за полем *Назва_товару*, поле *Дата* повинно стояти в бланку зліва від поля *Назва_товару*.

Щоб побачити результати роботи запиту, слід перейти з режиму конструктора в режим таблиці. Для цього служить кнопка  **Режим таблицы** на вкладці **Конструктор** стрічки.

Результатом запиту є тимчасовий набір даних (рис. 3.21), який формується в момент роботи запиту.

Будівлі_дані

Місто	Індекс	Назва_району	тип_вулиці	Назва_вулиці	Номер_будівлі	Номер_ЖЄД	Рік_будування
Харків	61001	Київський	вулиця	Пушкінська	5	2	1930
Харків	61168	Київський	вулиця	Героїв праці	12	12	1976
Харків	61168	Київський	вулиця	Блюхера	44	11	1978
Харків	60001	Дзержинський	проспект	Леніна	135	158	2010
Харків	61150	Дзержинський	проспект	Леніна	25	56	1950

Рис. 3.21 – Перегляд результату роботи запиту в режимі таблиці

При збереженні запиту зберігається не результат його роботи, а лише сама структура запиту в спеціальному форматі SQL. Щоб побачити запит у цьому форматі (рис. 3.22) слід використати кнопку  **Режим SQL** на вкладці **Конструктор** панелі **Результаты** стрічки.

Будівлі_дані

```
SELECT Будівлі.Місто, Будівлі.Індекс, Райони.Назва_району, Вулиці.тип_вулиці, Вулиці.Назва_вулиці,
Будівлі.Номер_будівлі, Будівлі.Номер_ЖЄД, Будівлі.Рік_будування, Матеріали.Назва_матеріалу,
Будівлі.Фото_будівлі, Будівлі.Кадастровий_код
FROM Райони INNER JOIN (Матеріали INNER JOIN (Вулиці INNER JOIN Будівлі ON Вулиці.Код_вулиці =
Будівлі.Код_вулиці) ON Матеріали.Код_матеріалу = Будівлі.Код_матеріалу) ON Райони.Код_району =
Будівлі.Код_району;
```

Рис. 3.22 – Запит у режимі SQL

У форматі SQL зберігаються і запити, створені за допомогою майстрів, і запити, створені в режимі конструктора. Конструктор запитів є лише зручним графічним інструментом для створення запитів за зразком QBE.

Редагування та виконання запитів

Редагування запитів

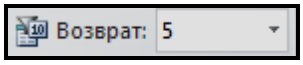
Для редагування запитів слід повернутися в режим конструктора і

внести зміни.

Для додавання в запит додаткових таблиць або запитів служить кнопка **Отобразить таблицу** 

на вкладці **Конструктор** панелі **Настройка запроса**. Для видалення зайвих таблиць і запитів слід виділити їх у верхній частині вікна і натиснути клавішу **Delete**.

Для видалення поля із запиту треба виділити це поле в бланку (нижній частині вікна) і натиснути клавішу **Delete**. Для виділення поля треба навести курсор миші над стовпцем поля так, щоб він набув форми чорної стрілки, і клацнути лівою кнопкою. Виділені поля можна переміщати в бланку, перетягуючи їх за верхню сіру смугу.

У випадку якщо потрібно отримати не повний, а частковий список результатів роботи запиту (наприклад, перші 10 записів або 5 % усіх записів), можна на панелі інструментів вказати необхідне значення за допомогою кнопки **Возврат** 

на вкладці **Конструктор** панелі **Настройка запроса**. При цьому дозволяється не тільки вибирати значення з випадаючого списку цієї кнопки, але й вводити свої, довільні значення.

Наприклад, можна для запиту, наведеного попереду, встановити у полі *Назва_товару* сортування за убуванням цін і вказати у списку **Возврат –** 1. Це означатиме вимогу вивести один запис результуючого набору значень, тобто найдорожчий товар.

У режимі конструктора запитів можна змінювати імена полів запиту. Щоб перейменувати поле, необхідно встановити курсор у бланку запиту перед першою буквою імені поля і ввести нове ім'я та двокрапку. Нове ім'я поля буде відображатися як заголовок стовпця при прогляданні запиту в режимі таблиці. Крім того, у формах, звітах і тому подібне, заснованих на даному запиті, також буде використане нове ім'я поля. Ім'я поля базової таблиці при цьому не змінюється.

Для форматування даних якого-небудь поля слід викликати вікно властивостей цього поля (команда 

на вкладці **Конструктор** панелі **Показать или скрыть**) і заповнити властивість **Формат поля** та **Число десятичных знаков**.

Виконання запитів

Запустити запит на виконання можна різними способами – як явними, так і неявними. До неявних відноситься використання запитів у фор-

мах, інших запитах, звітах, формування списків підстановки таблиць і т. п.

Для явного запуску запиту у вікні БД на панелі **Все объекты** MS Access слід двічі клацнути мишею на назві запиту або виділити його і в контекстному меню вибрати команду **Открыть**.

Для запуску запиту з режиму конструктора слід використовувати кнопку 

Режим таблицы на вкладці **Конструктор** стрічки.

Розширені можливості запитів

Запити, що засновані на даних декількох таблиць

Якщо запит заснований на даних відразу декількох таблиць або запитів, треба додати їх за допомогою вікна вибору таблиць та запитів і прослідкувати, щоб вони були з′єднані. Якщо зв'язки між таблицями в базі даних уже визначені, лінії з′єднання відображаються автоматично. Якщо таблиці, що беруть участь у запиті, не пов'язані між собою безпосередньо, необхідно додати одну або декілька додаткових таблиць, які пов'язують вибрані раніше таблиці.

Так, для побудови запиту, що відображає дані про квартири з їхньою адресою, необхідно включити у запит таблицю *Квартири*. Проте ця таблиця безпосередньо не пов'язана з таблицями *Райони*, *Вулиці* та *Матеріали* в базі даних. При виконанні такого запиту кожній квартирі будуть приписані дані кожного наявного запису таблиць-довідників. Щоб відобразити реальний стан справ, потрібно за допомогою вікна схеми даних з'ясувати, які таблиці пов'язують таблицю *Квартири* з таблицями *Райони*, *Вулиці* та *Матеріали*, і додати їх у запит. Для даного прикладу бази даних має бути додатково додана таблиця *Будівлі* (рис. 3.23).

Якщо в запит були додані не пов'язані між собою таблиці, які мають поля з однаковими іменами і узгодженими типами даних, конструктор запитів автоматично встановить зв'язки між цими полями таблиць. Найчастіше це приводить до некоректних результатів запиту. Наприклад, якщо в запит додати дві непов'язані таблиці, що містять поля *Назва* (припустимо, назви вулиць і назви районів міста), MS Access пов'яже таблиці за цими двома полями, що неправильно.

Тому необхідно перевіряти зв'язки між таблицями у вікні конструктора. Зайві зв'язки треба видаляти, виділивши лінію зв'язку і натиснувши клавішу **Delete**.

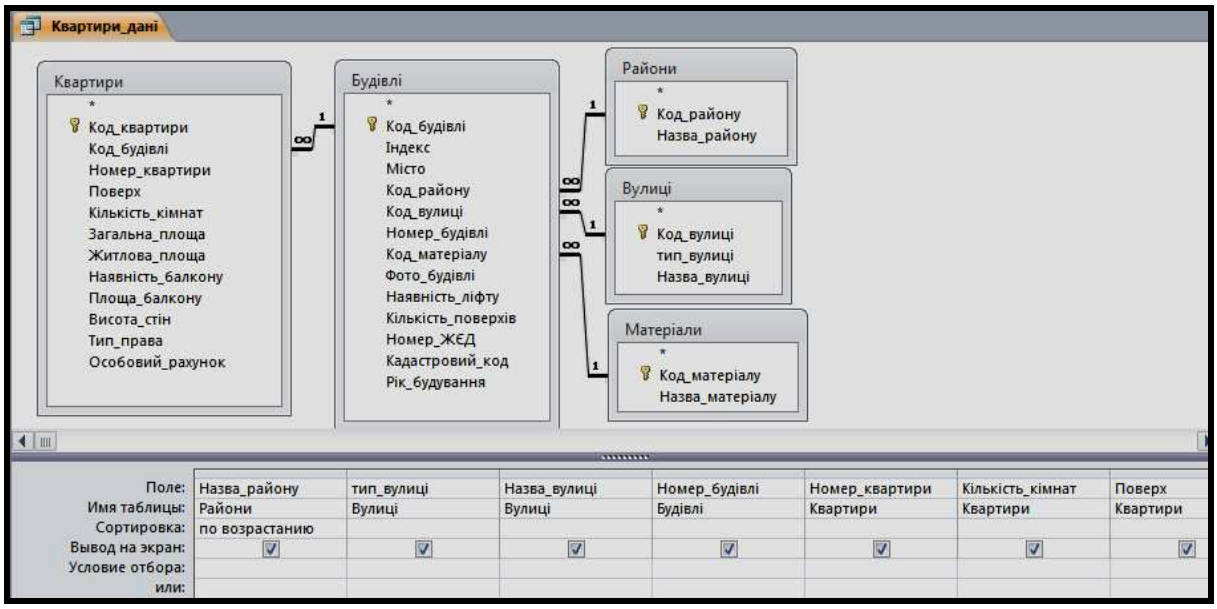


Рис. 3.23 – Запит, заснований на даних декількох таблиць

Параметричні запити

Запит у MS Access зберігається у файлі бази даних і може багато разів повторюватися. Усі створені до цих пір запити містили конкретні значення дат, назв та імен. Умови вибору можна змінити в режимі конструктора. Щоб не виконувати ці операції багато разів, можна створити запит із параметрами. При виконанні такого запиту з'являється діалогове вікно **Введите значение параметра**, в якому користувач може ввести конкретне значення й отримати потрібний результат.

Припустимо, що потрібно вивести дані по квартирах з певною кількістю кімнат. Для цього вкажемо замість кількості кімнат параметр. З цією метою замінимо умову відбору в полі *Кількість_кімнат* на довільну фразу-запрошення, розташовану у квадратних дужках, наприклад [*Вкажіть кількість кімнат*] (рис. 3.24).

Під час виконання запиту на екран буде виведено вікно, що містить вписану фразу-запрошення і рядок введення (рис. 3.25). Після введення будь-якої кількості кімнат ця кількість буде підставлена в умову вибору замість параметра, і запит буде виконаний для цього значення.

Параметри можуть використовуватися в умовах вибору спільно з логічними або арифметичними операторами та операторами порівняння. Можна також застосовувати в одному виразі декілька параметрів.

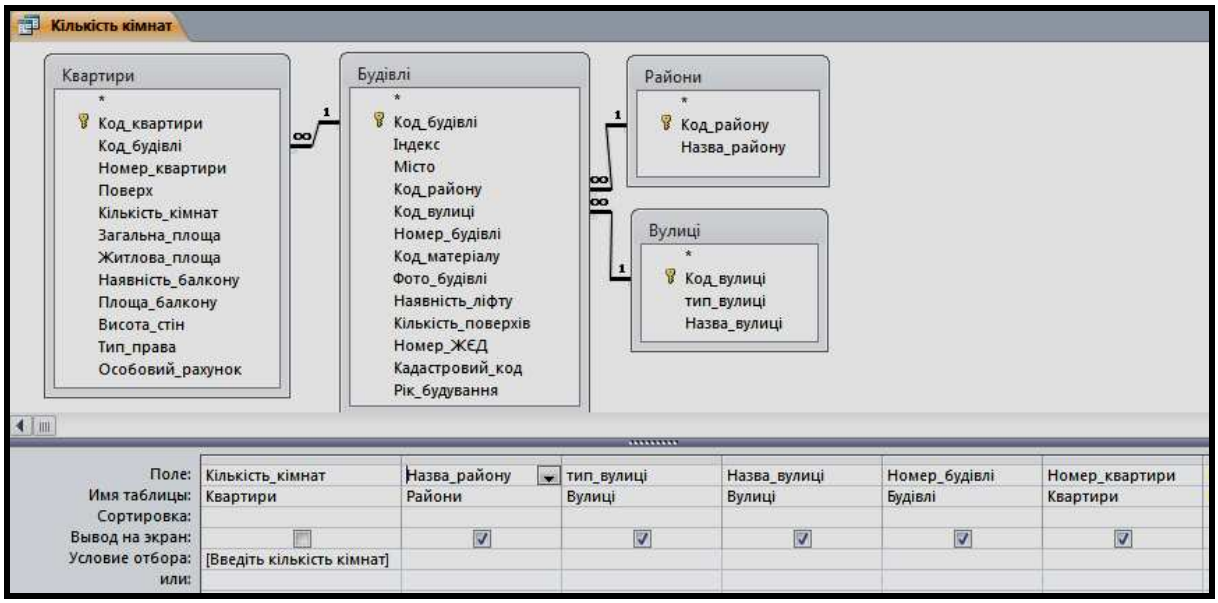


Рис. 3.24 – Використання параметра в запиті

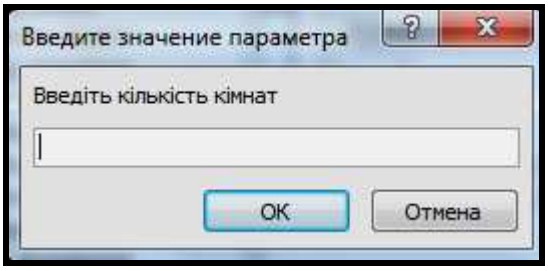


Рис. 3.25 – Вікно введення значення параметра

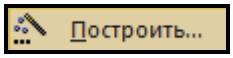
Обчислення в запитах

Створення обчислюваних полів у запитах

Створення запитів цікаве не тільки тим, що можна відібрати потрібні записи з декількох пов'язаних таблиць і представити дані з них у вигляді однієї таблиці. Можна створювати стовпці в запиті, які є результатом обчислень над значеннями інших полів. Такі поля називаються обчислюваними. Це істотно розширює можливості запитів.

Наприклад, можна проводити розрахунок нежитлової площі квартир (комори, балкони та ін.), визначати житлові площі за окремими будинками та цілими районами, об'єднувати значення декількох полів в одному полі, обчислювати вартість квартир і т. д.

Нескладну формулу можна ввести у вільне поле бланка запиту прямо в рядку **Поле**. Вирази можна вводити в середовищі MS Access не тільки вручну, а й за допомогою інструменту **Построитель выражений**. Для його виклику служить команда контекстного меню **Построить**



або однойменна команда на панелі **Настройка запроса** вкладки **Конструктор** стрічки.

Верхня частина вікна **Построитель выражений** призначена для введення формули, нижня частина вікна полегшує процес введення формул. Розташовані в центрі вікна кнопки допомагають набирати знаки і оператори за допомогою миші.

Щоб швидко вставити у формулу назву поля іншої таблиці (запиту) або функцію, треба розкрити за допомогою кнопки «+» необхідну категорію в лівому стовпці, вибрати назву поля у середньому стовпці (для функцій – у крайньому правому) і здійснити подвійне клацання мишею (рис. 3.26).

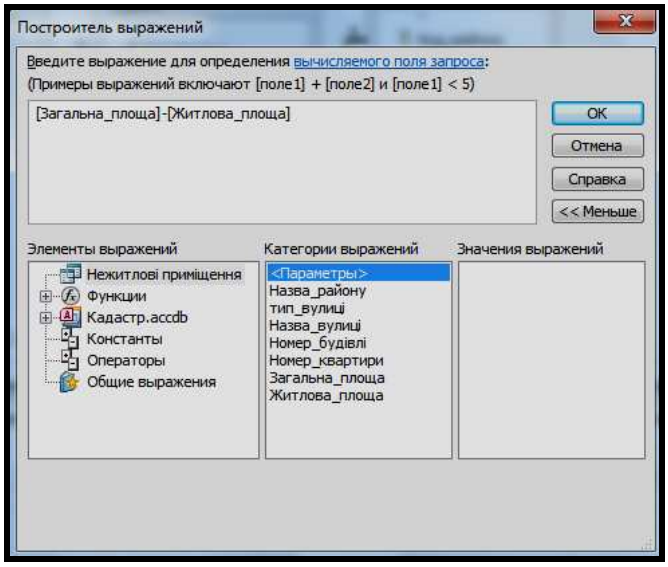


Рис. 3.26 – Вікно **Построитель выражений** для введення виразів

Аналіз даних за допомогою запитів

Запити з групуванням даних

MS Access надає можливість групувати записи в результуючому наборі даних запиту. Механізм групування даних дозволяє як прибрати записи, що повторюються, так і провести обчислення над кожною групою запису.

Для організації групування необхідно в режимі конструктора скористатися командою Σ Итоги **Итоги** на панелі **Показать или скрыть** вкладки **Конструктор** стрічки або вибрати цю команду в контекстному меню. У бланку запиту з'явиться додатковий рядок **Групповая операция**. За замовчуванням у цьому рядку для кожного поля буде встановлено значення **Группировка**.

Для проведення обчислень над деяким полем групи записів треба в

рядку **Групповая операция** вибрати одну з таких статистичних функцій: **Sum**, **Avg**, **Min**, **Max**, **Count**, **First**, **Last**, **StDev** або **Var**. Опис функцій наведено в табл. 3.2.

Таблиця 3.2 – Статистичні функції в рядку «Групповые операции»

Назва функції	Результат, що повертається
Sum	Сума значень поля
Avg	Середнє від значень поля
Count	Число значень поля без урахування порожніх (Null) значень
Min	Найменше значення поля
Max	Найбільше значення поля
Var	Дисперсія значень поля
StDev	Середньоквадратичне відхилення від середнього значення поля
First	Значення поля з першого запису результуючого набору
Last	Значення поля з останнього запису результуючого набору

Слід ураховувати, що при обчисленні статистичних функцій MS Access не враховує записи, що містять порожні значення (**Null**). Наприклад, функція **Count** повертає кількість всіх не порожніх полів (що не містять значення **Null**).

Приклад. Щоб розрахувати сумарну житлову площу за районами міста, треба згрупувати записи за полем **Назва_району**, а для поля **Житлова_площа** встановити розрахунок за функцією **Sum** (рис. 3.27).

MS Access присвоює полю з підсумковою функцією ім'я, створене шляхом об'єднання імені функції та імені поля (**Sum-Житлова_площа**). Для перейменування поля слід у режимі конструктора запиту встановити курсор у першу позицію рядка **Поле**, ввести нову назву і двокрапку. Будьте акуратні, не зіпсуйте колишню назву поля, для якого проводиться групова операція.

Якщо результуючий вираз достатньо складний, MS Access переносить статистичну функцію в рядок **Поле**, замінюючи її назву в рядку **Групповая операция** на елемент **Выражение**.

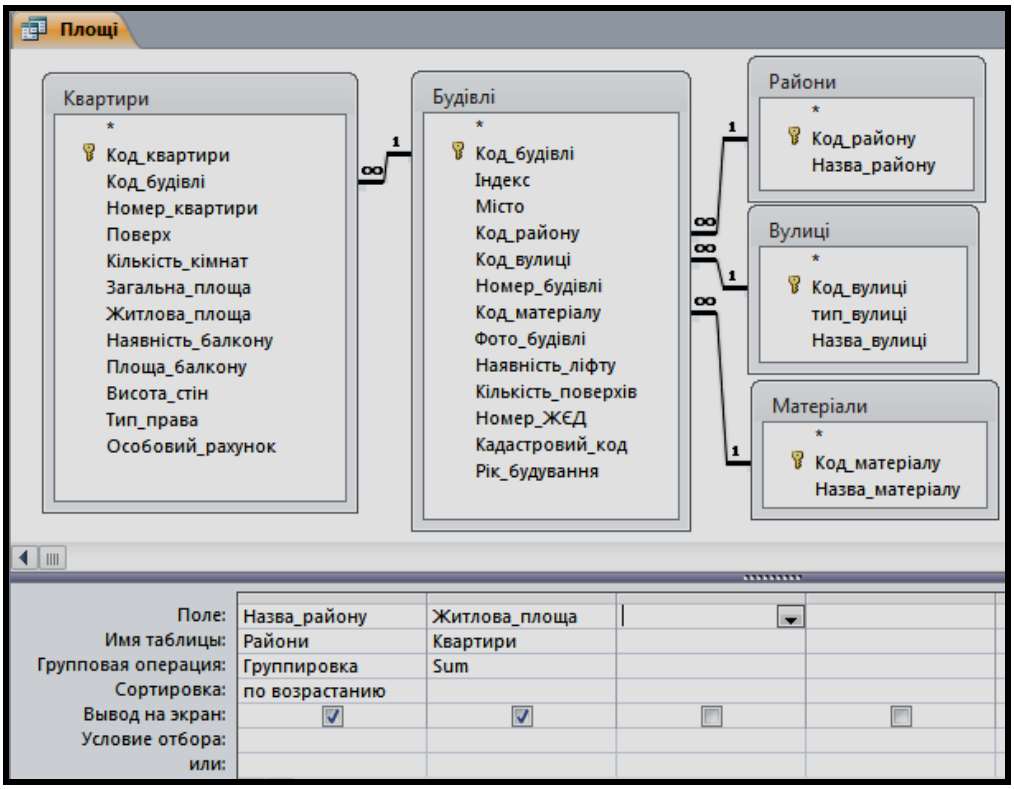


Рис. 3.27 – Запит із групуванням даних

У тому випадку, коли які-небудь поля заважають провести групування даних, слід вказати **Условие** в рядку **Групповая операция**. При цьому автоматично буде знятий прапорець виведення поля на екран, і поле не виводитиметься при виконанні запиту.

Перехресні запити

Перехресні запити мають такі переваги:

- можливість обробки значного об'єму даних і виводу їх у форматі, відповідному для автоматичного створення графіків та діаграм;
- простота і швидкість розробки складних запитів за декількома рівнями деталізації.

Проте вони мають і недоліки, наприклад, не можна сортувати таблицю результатів за даними, що містяться у стовпцях. При цьому можна задавати сортування за збільшенням або за убуванням для заголовків рядків.

Для створення перехресного запиту можна скористатися майстром або побудувати його самостійно в режимі конструктора. У першому випадку належить спочатку створити в конструкторі звичайний запит на вибірку даних, що відбирає поля для майбутнього перехресного запиту. У другому випадку процес створення може бути здійснений за один крок.

Створення перехресного запиту за допомогою майстра

Розглянемо завдання отримання інформації про сумарну житлову площу помешкань за районами міста. Організуємо перехресний запит так, щоб в рядках містилися назви районів, заголовками стовпців служили назви вулиць міста, а на перетині рядків і стовпців розташовувалися значення сумарної площі квартир.

Спочатку слід створити запит на вибірку, що містить усі необхідні поля, в тому числі і поле *Житлова площа*.

Створений запит слід зберегти. Потім викликати майстер створення запитів за допомогою кнопки 

Мастер запросов панелі **Запросы** вкладки **Создание** стрічки та вибрати спосіб створення **Перекрестный запрос**.

На першому кроці майстра вибрати запит, у якому містяться всі необхідні поля, в тому числі і розрахункові.

На другому кроці майстра у відповідь на запитання, яке поле буде використано як заголовки рядків, вказати поле *Назва_району*.

На третьому кроці у відповідь на запитання, яке поле буде використано як заголовки стовпців, треба вибрати поле *Назва_вулиці*.

Результат кожного покрокового уточнення майбутнього запиту відображається в нижній частині вікна у зразку.

На четвертому кроці виберіть статистичну операцію над груповаными даними – відповідно до умови це має бути операція **Сумма** для поля *Житлова_площа*. MS Access з'ясовує, чи слід підбивати підсумки за кожним рядком – для цього призначений відповідний прапорець опцій у вікні.

Вивівши запит у режимі конструктора (рис. 3.28), можна побачити, що в бланку запиту доданий рядок **Перекрестная таблица**. У цьому рядку для кожного поля вказано його місцеположення при виведенні результатів (заголовки рядків, заголовки стовпців або значення на перетині рядків і стовпців таблиці). Остаточний вигляд перехресного запиту в режимі таблиці наведено на рис. 3.29.

Побудова запитів на підставі запитів

Багато завдань важко, а часом неможливо вирішити за один крок. У таких випадках доводиться розробляти серію запитів, заснованих один на іншому.

Розглянемо завдання обліку документів що, підтверджують право власності на помешкання. Створимо для цієї мети запит із полями *Квар-*

тиронаймач із таблиці *Мешканці* і *Код_квартири*, *Вид_договору*, *Номер_документу*, *Дата_реєстрації*, *Номер_у_книзі_записів* із таблиці *Документи*. Окрім названих таблиць, у запит належить включити і таблиці, що їх пов'язують – *Квартири*, а також *Будівлі* і *Вулиці* – для визначення адреси помешкання.

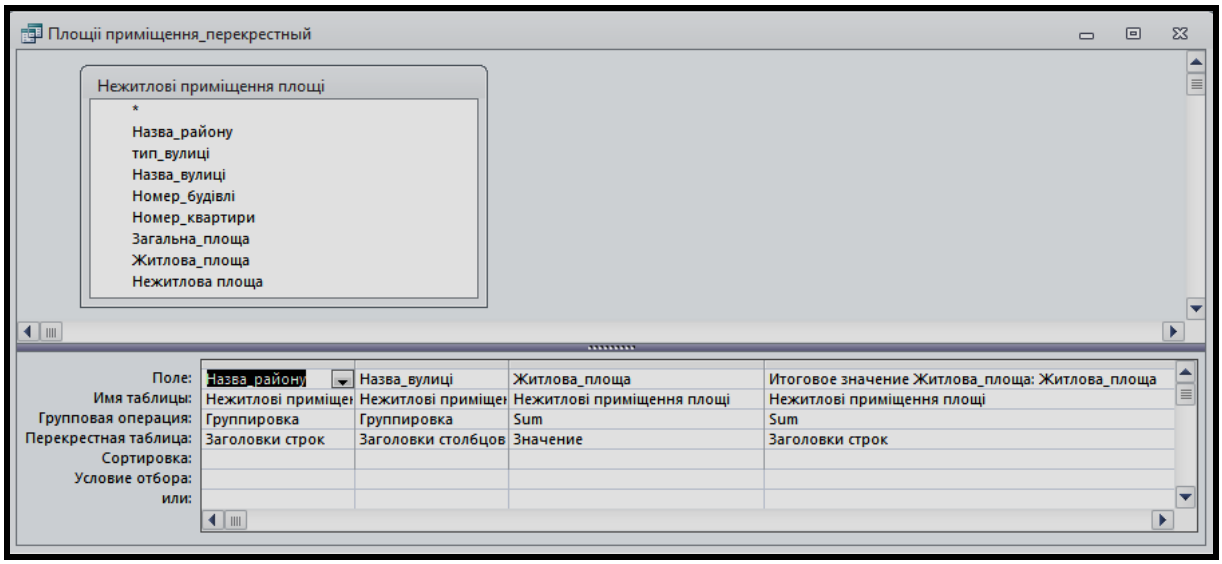


Рис. 3.28 – Вигляд перехресного запиту в режимі конструктора

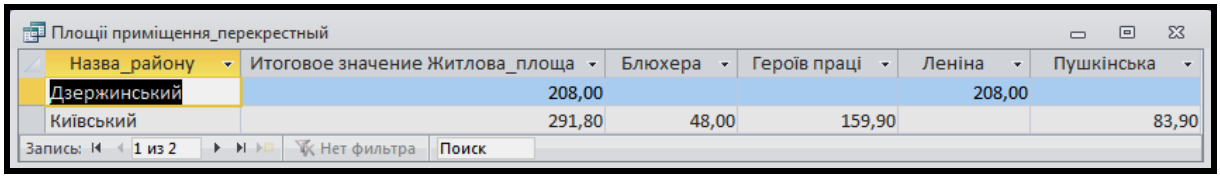


Рис. 3.29 – Вигляд перехресного запиту в режимі таблиці

Якщо підрахувати кількість записів, то відповідь значно перевищить реальну кількість квартир. Це відбудеться тому, що у кожній квартирі прописано по декілька мешканців. Щоб уникнути такої ситуації, на основі тільки що створеного запиту створимо ще один запит. До властивості **Условие отбора** поля *Квартиронаймач* слід додати умову: **Истина**, для того щоб запит показував тільки дані, пов′язані з відповідальним квартиронаймачем, а такий у кожному помешканні тільки один. Тепер можна додавати до запиту необхідні поля, оскільки ми ліквідували значення, що повторюються (рис. 3.30). Запит, на базі якого створений наступний, називається підлеглим запитом.

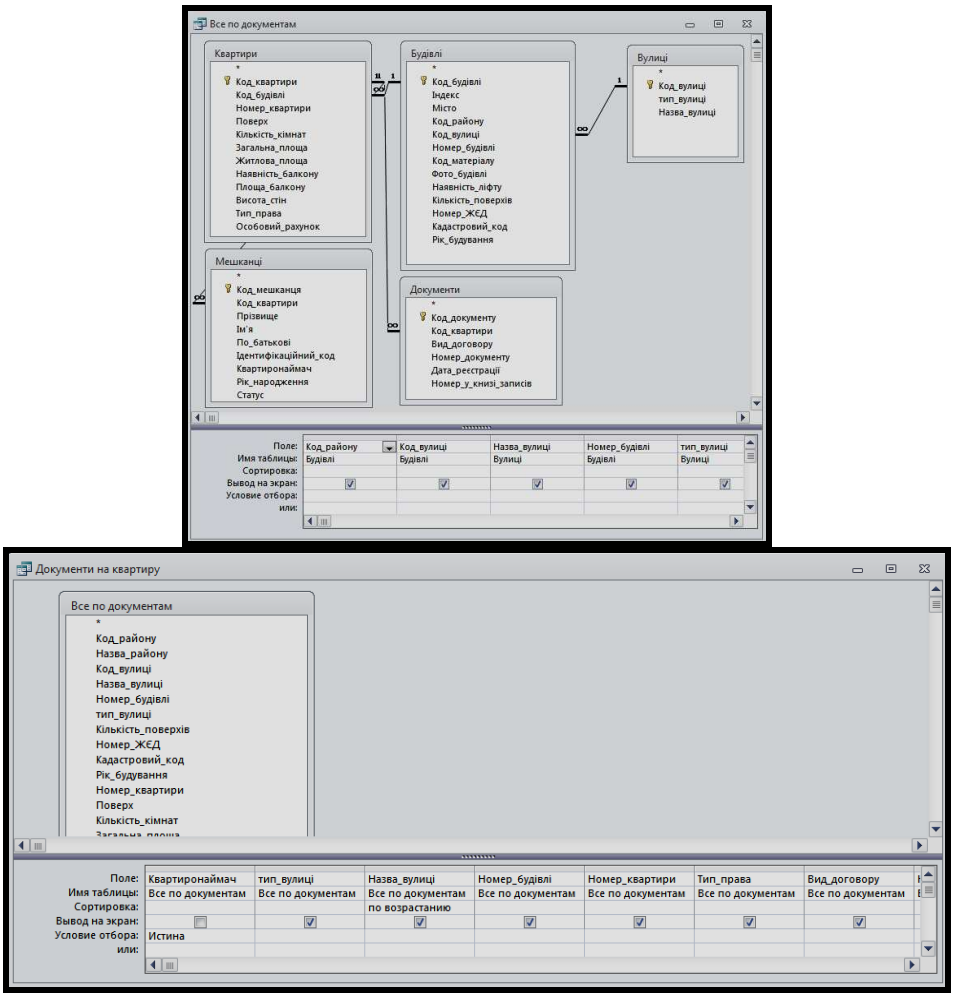


Рис. 3.30 – Створення запиту на базі іншого запиту

Види з'єднань між таблицями в запитах

З'єднання між таблицями в запитах може бути декількох видів. Найбільш поширеним є внутрішнє з'єднання. Якщо таблиці пов'язані відношенням «один-до-багатьох», з'єднання ґрунтуються на унікальному значенні поля первинного ключа в одній таблиці і значеннях поля зовнішнього ключа в іншій таблиці. Подібного роду з'єднання між таблицями MS Access створює автоматично, якщо:

- з'єднання було явно задане у вікні **Схема данных**;
- у таблицях є поля з однаковими іменами й узгодженими типами, причому одне з полів є ключовим.

Результатом такого запиту є всі записи, значення пов'язаних полів яких в обох таблицях збігаються. У результуючу множину запиту потрапляють усі записи з головної таблиці (на стороні «один»), для яких є відповідні записи в підлеглій таблиці (на стороні «багато»). Якщо в підлеглій таблиці записи із заданою величиною відсутні, то відповідні записи в головній

таблиці у результуючу множину не включаються.

Для створення запиту, що об'єднує всі записи з однієї таблиці і лише ті записи з другої, в яких пов'язані поля збігаються, використовують зовнішнє з'єднання. У цьому випадку незалежно від того, чи є відповідні записи в другій таблиці, всі записи першої потрапляють у створений запит.

Щоб змінити внутрішнє з'єднання таблиць на зовнішнє, слід у режимі конструктора запитів клацнути двічі на лінії, що об′єднує таблиці, або вибрати з її контекстного меню команду **Параметры объединения**. У вікні, що відкрилося (рис. 3.31), буде відображено три варіанти з'єднання таблиць – одне внутрішнє і два зовнішніх. За замовчуванням встановлено внутрішнє з'єднання.

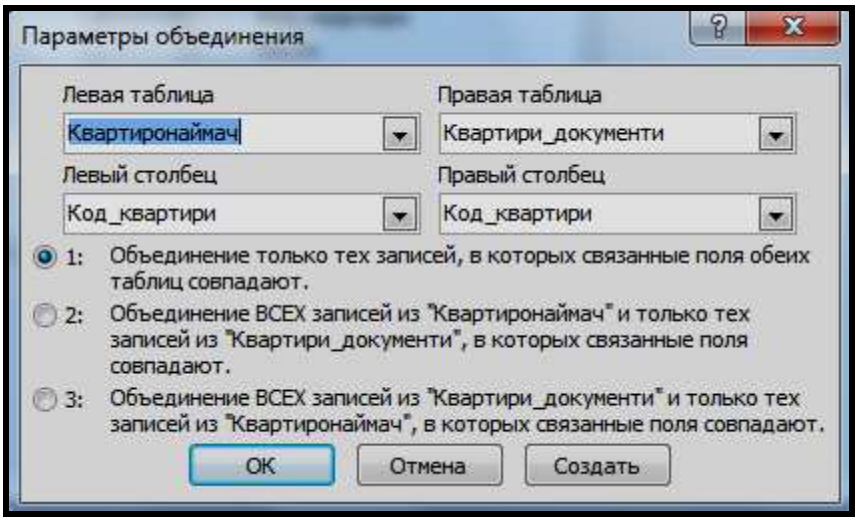


Рис. 3.31 – Види з'єднань між таблицями в запиті

Для зв′язування даних в одній таблиці застосовують також рекурсивне з'єднання. Воно створюється шляхом додавання таблиці в запит двічі, внаслідок чого MS Access призначає псевдонім для копії.

При використанні рекурсивних з'єднань потрібно задати вивід тільки унікальних значень. Для цього у вікні **Свойства запроса** слід встановити значення **Да** для властивості **Уникальные значения**.

Аналогічно рекурсивному з'єднанню можуть бути встановлені з'єднання між довільними полями таблиць чи запитів для виявлення нових зв'язків між даними. Наприклад, для вирішення завдання з контролю за прописаними квартиронаймачами у помешканнях створюється запит на підставі вже існуючих запитів. На рис. 3.32 показано приклад зміненого зв'язку між запитами, на підставі яких створено новий запит. У схемі даних ці

запити з′єднані зв'язком типу «один-до-одного» по полю *Код_квартири*.

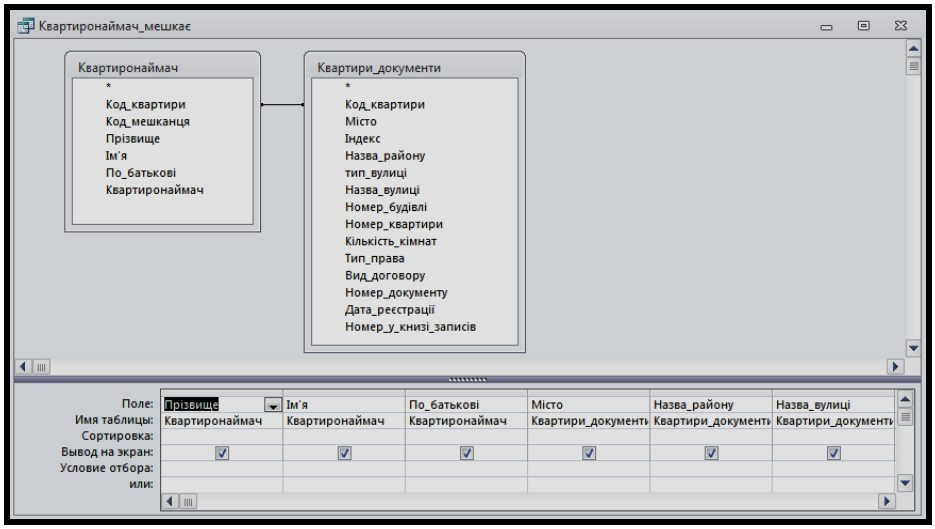


Рис. 3.32 – Зміна зв'язку між таблицями

Модифікуючі запити

Існує чотири типи модифікуючих запитів, які виконують дії над записами таблиці:

- запити створення таблиці;
- запити додавання записів у таблицю;
- запити видалення записів із таблиці;
- запити оновлення записів таблиці.

Розглянемо ці варіанти запитів.

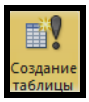
Запити створення таблиць

Процес створення таблиці за допомогою запиту складається із трьох кроків:

1. Створення запиту на вибірку.
2. Перетворення запиту на вибірку в запит на створення, задавши параметри розміщення нової таблиці.
3. Виконання запиту на створення, тим самим помістивши відібрані записи в нову таблицю.

Класичним прикладом такого запиту є створення з його допомогою нової таблиці *Діти війни* в разі обліку таких громадян. Щоб створити такий запит, необхідно:

- створити запит на вибірку, який містить необхідні поля і використовується для відбору записів. Ввести необхідні умови відбору (у полі *Рік_народження* – умова *<=1945*);

- перетворити його на запит на створення таблиці (команда **Создание таблицы** 

на панелі **Тип запроса** вкладки **Конструктор**);
- проглянути оновлені записи в режим таблиці.

У разі виконання запиту MS Access видає повідомлення про кількість розміщених записів і запрошує підтвердження на створення нової таблиці (рис. 3.33).

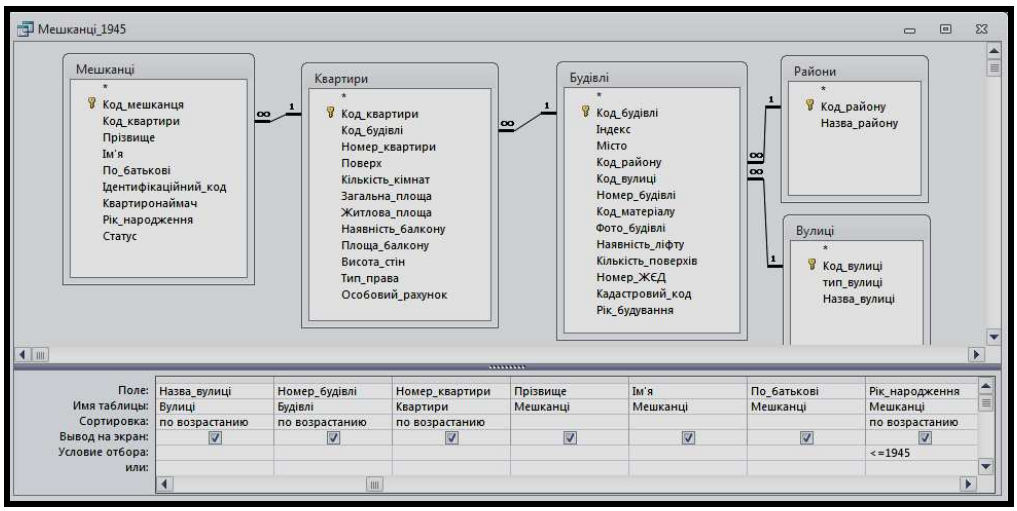


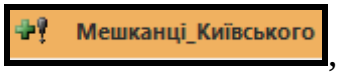
Рис. 3.33 – Запит на створення нової таблиці

Запити на додавання записів у таблицю

Для додавання записів у таблицю треба створити запит на вибірку, що містить необхідні поля, перейти в режим конструктора та встановити тип запиту **Добавление** за допомогою кнопки 

на панелі **Тип запроса** вкладки **Конструктор**. У вікні **Добавление** виберіть ім'я тієї таблиці, в яку додаються записи. У бланку запиту з'явиться додатковий рядок **Добавление**. Він містить назви полів таблиці, в яку додаються записи. Потім помістіть у бланк запиту поля, які необхідно додати, а також ті, які будуть використані для умов вибору.

Після вказівки умов вибору слід проглянути запит у режимі таблиці і тільки потім запускати його на виконання.

Зверніть увагу на значок в імені запиту 

, що відповідає запиту на додавання.

Запити на видалення записів із таблиці

Процес видалення записів із таблиці за допомогою запиту складаєть-

ся із трьох кроків:

1. Створення запиту на вибірку на базі однієї таблиці.
2. Перетворення запиту на вибірку в запит на видалення.
3. Виконання запиту на видалення, тим самим видаливши відібрані записи з таблиці.

Запити на оновлення записів таблиці

Запити на оновлення використовуються головним чином для того, щоб внести зміни відразу у велику кількість записів за допомогою одного запиту. Класичним прикладом такого запиту є зміна тарифів на послуги в разі їх обліку у базі даних.

Прикладом такого запиту є змінення *Ціна_м_кв* в таблиці *Додаткова нерухомість* в разі підвищення тарифів. Можна збільшити ціну на додаткову нерухомість, наприклад на 5 %. Щоб створити такий запит, необхідно:

1. Створити запит на вибірку, який містить поля, що підлягають оновленню й використовуються для відбору записів. Ввести необхідні умови відбору.
2. Перетворити його на запит на оновлення (команда **Обновление**  на панелі **Тип запроса** вкладки **Конструктор**). При цьому з'являється додатковий рядок **Обновление**, а також зникають рядки **Сортировка** та **Вывод на екран**.
3. Ввести в рядок **Обновление** вираз, за яким буде проводитися обчислення.
4. Проглянути оновлені записи в режим таблиці. Звернути увагу на те, що таблиця, яка виводиться, містить тільки ті поля, для яких у рядку **Обновление** було введено вираз або задане **Условие отбора**.

У разі виконання запиту, так само як і при додаванні записів у таблицю, MS Access видає повідомлення про кількість оновлюваних записів і запрошує підтвердження на оновлення.

Особливим є випадок, коли відновлюються значення первинного ключа таблиці. Якщо ця таблиця пов'язана відношенням «один-до-багатьох» з іншими таблицями, то при зміні первинного ключа запису одночасно повинні змінитися й значення зовнішніх ключів у всіх пов'язаних записах підлеглих таблиць. MS Access забезпечує виконання такої зміни автоматично, оскільки підтримує каскадне оновлення записів.

Якщо таке оновлення встановлене, то при зміні значення первинного ключа в головній таблиці MS Access автоматично виконує спеціальний запит, за допомогою якого оновлюються зовнішні ключі всіх пов'язаних записів у підлеглій таблиці.

Створення запитів в режимі SQL Access

Мова SQL призначена для створення бази даних і таблиць, а також для маніпулювання даними за допомогою операцій додавання, видалення і модифікації. Запити SQL створюються в режимі SQL і використовуються для роботи з таблицями бази даних, а також для створення підпорядкованих запитів безпосередньо в рядках **Поле** або **Условия выбора** в бланку запиту конструктора.

У склад мови SQL входять два основні компоненти:

мова DDL (Data Definition Language) – використовується для опису структур баз даних і управління доступом до них;

мова DML (Data Manipulation Language) – використовується для вибірки й оновлення даних.

MS Access не надає користувачеві командного рядка для безпосереднього введення SQL-конструкцій. Для цього використовується вікно конструктора запитів. Алгоритм роботи такий:

1. Відкрийте вкладку **Создание** на стрічці з вкладками MS Access.
2. Виберіть панель **Запросы** та команду 

Конструктор запросов.
3. З'явиться вікно **Добавление таблицы**. Закрийте його.
4. Виберіть команду 

Режим SQL панелі **Результаты** .
5. Відкриється вікно конструктора в режимі введення кода мови SQL.
6. Введіть у вікні текст запиту і збережіть запит.
7. Відкрийте запит і перевірте результат його виконання.

Визначення даних за допомогою мови DDL

У склад мови DDL входять чотири оператори:

CREATE TABLE – створення нової таблиці;

ALTER TABLE – зміна структури таблиці;

DROP – видалення таблиці або індексу;

CREATE INDEX – створення індексу.

Вони дозволяють автоматизувати роботу зі створення, модифікації й видалення таблиць без використання конструктора.

Створення таблиць за допомогою мови DDL

Оператор **CREATE TABLE** визначає ім'я таблиці і множину поіменованих стовпців у вказаному порядку. Для кожного стовпця має бути визначений тип і розмір.

Синтаксис оператора:

CREATE TABLE *<Ім'яТаблиці>* (

[<ІмяПоля1> <ТипДаних1> [<Розмір1>] [**CONSTRAINT** <Індекс1>]

[<ІмяПоля2> <ТипДаних2> [<Розмір2>] [**CONSTRAINT** <Індекс2>])

Тут і далі великими буквами виділені ключові слова. У кутових дужках курсивом позначені складові елементи оператора, наприклад:

<Розмір1>, *<Розмір2>* – розмір поля в знаках (тільки для полів із типом даних TEXT, BINARY і STRING);

<Індекс1>, *<Індекс2>* – індекс за одним полем, який визначає параметр **CONSTRAINT** (обмеження).

У квадратних дужках розміщені необов'язкові елементи.

Типи даних, які можуть бути створені за допомогою оператора **CREATE TABLE** в MS Access 2010, наведені в табл. 3.3.

Таблиця 3.3 – Типи даних мови SQL MS Access

Тип даних	Опис
BYTE	**Байт**
SMALLINT або **SHORT**	**Целое**
INTEGER або **LONG**	**Длинное целое**
SINGLE	**Число с плавающей точкой одинарной точности**
NUMERIC, FLOAT або **DOUBLE**	**Число с плавающей точкою двойной точности**
CHAR(N), **TEXT**(N) або **STRING** (N)	**Текстовый – длиной N символов**
BINARY(N)	**Действительное (N разрядов)**
DATE або **DATETIME**	**Дата/время**
CURRENCY	**Денежный**
LOGICAL або **BIT**	**Логический**

Наступні приклади показують створення таблиць «*Будівлі*» і «*Вулиці*» бази даних **«*Кадастр*»**.

Приклад. Запит на створення таблиці «***Вулиці***».

CREATE TABLE *Вулиці* (*Код_вулиці* LONG CONSTRAINT *Код_вулиці* PRIMARY KEY,

Тип_вулиці TEXT(15),

Назва_вулиці TEXT(30))

Другий рядок запиту призначений для створення первинного ключа (*Код_вулиці* **PRIMARY KEY**). Наступні рядки призначені для створення полів *Тип_вулиці* та *Назва_вулиці*. Вони мають тип текстовий.

Приклад. Запит на створення таблиці «***Будівлі***».

CREATE TABLE *Будівлі* (

Код_будівлі LONG CONSTRAINT *Код_будівлі* PRIMARY KEY,

Індекс INTEGER,

Місто TEXT(20),

Код_вулиці LONG CONSTRAINT *Код_вулиці* REFERENCES *Вулиці*,

Номер_будівлі TEXT(5),

Наявність_ліфту LOGICAL,

Номер_ЖЕД INTEGER,

Рік_будування INTEGER,

Кадастровий_код TEXT(15))

Другий рядок запиту призначений для створення первинного ключа (*Код_будівлі* PRIMARY KEY). П′ятий рядок створює стовпець *Код_вулиці*, який призначений зовнішнім ключем (CONSTRAINT *Код_вулиці*) для зв'язку (REFERENCES) з таблицею «***Вулиці***».

Для створення індексу зовнішнього ключа *Код_клієнта* використаємо оператор **CREATE INDEX**.

Приклад. Запит на створення індексу поля Код_вулиці таблиці «Будівлі».

CREATE INDEX *Будівлі*

ON *Будівлі* (*Код_вулиці*)

Для створення індексу по одному полю зовнішнього ключа ***Код_вулиці*** в круглих дужках після імені таблиці «***Будівлі***» вказано ім'я цього поля.

Після виконання запитів у базі даних «***Кадастр***» створюються таблиці «***Вулиці***», «***Будівлі***» та зв'язки між ними (див. рис. 2.18).

Створення запитів за допомогою мови DML

Оператор SELECT

Оператор **SELECT** призначений для вибірки необхідних рядків із бази даних і розміщення їх у динамічному об'єкті − наборі записів.

Синтаксис оператора:

SELECT <Список_стовпців>
FROM <Список_таблиць>
[**WHERE** <Умова_вибору>]
[**GROUP BY** <Список_полів_групи]
[**HAVING** <*Умова*>]
[**ORDER BY** <Список_атрибутів>]**;**

Ключове слово **SELECT** (вибір) визначає імена полів таблиці, яку створює запит. Ключове слово **FROM** визначає імена таблиць, які є джерелом записів для створюваного запиту.

Приклад. Для вибірки всіх стовпців таблиці «***Вулиці***» бази даних «***Кадастр***» необхідно написати:

SELECT *Тип_вулиці, Назва_вулиці*
FROM *Вулиці*;

Точку з комою в кінці останнього рядка MS Access додає сам.

Ключове слово **WHERE** дозволяє задавати вираз умови, що набуває значення **True** або **False** для значень полів таблиць, до яких звертається оператор **SELECT**. У запит включаються тільки ті рядки, для яких умова, вказана в пропозиції **WHERE**, набуває значення «істина».

Текст запиту, що виконує вибірку тих будівель, що розташовані на вулиці «***Пушкінська***» (код 1) із таблиці «***Будівлі***», такий:

SELECT *Код_вулиці, Номер_будівлі, Рік_будування, Кадастровий_код*
FROM *Будівлі*
WHERE *Код_вулиці* = 1
ORDER BY *Номер_будівлі*;

Ключові слова **ORDER BY** призначені для впорядкування даних. Вони упорядковують виведення запиту згідно із значеннями в тій або іншій кількості вибраних стовпців. Можна використовувати **ORDER BY** одночасно з будь-яким числом стовпців. У всіх випадках стовпці, які упорядковуються, мають бути вказані у ключовому слові **SELECT**. За замовчуванням встановлено зростання. Текстові поля будуть відсортовані в алфавітному порядку. Стовпці типу **Дата/час** – в хронологічному порядку.

Для встановлення сортування за убуванням використовується оператор **DESC**, наприклад, оператор

ORDER BY *Номер_будівлі* DESC

виконає сортування поля *Номер_будівлі* за убуванням.

Ключові слова **GROUP BY** призначені для об'єднання записів з однаковими значеннями, що знаходяться у вказаному списку полів, в один запис.

Ключове слово **HAVING** (наявність) визначає згруповані записи, які повинні відображатися в операторі **SELECT** з реченням **GROUP BY**. Після того як записи будуть згруповані, **HAVING** покаже ті з них, які відповідають його умовам.

Приклад. Для пошуку кількості будівель, що розташовані на вулиці, необхідно написати:

SELECT *Код_вулиці*, Count (*Номер_будівлі*) AS [Count-*Номер_будівлі*]
FROM *Будівлі*
GROUP BY *Код_вулиці*
HAVING *Код_вулиці*=4;

У запиті використовується функція **Count** (). Вона підраховує кількість будівель на вулиці. Аргументом є строковий вираз. Операндом у виразі може бути ім'я таблиці або функція.

Вибірка даних із декількох таблиць

При роботі з нормалізованою базою даних неодмінно потрібно буде створювати запити, що використовують дані з декількох таблиць. Їх з'єднання обов'язкове. Для з'єднання таблиць використовуються ключові слова **INNER**, **JOIN**, **ON** (**JOIN –** операція з'єднання).

Приклад. Для вибору мешканців, що народилися за останній рік, з таблиці «***Мешканці***» бази даних «***Кадастр***» необхідно написати:

SELECT *Мешканці.Код_мешканця*, *Мешканці.Рік_народження*

FROM *Мешканці*

WHERE (((*Мешканці.Рік_народження*) **Between** *#1/1/2012#* **And** *#31/12/2012#*))**;**

У цьому прикладі для задання інтервалу дат використовуються оператори **Between** та оператор **And**.

Приклад. Для вибірки даних щодо розташування будівель по вулицях із таблиць «***Вулиці***» та «***Будівлі***» необхідно написати:

SELECT *Вулиці.Назва_вулиці*, *Будівлі.Номер_будівлі*

FROM *Вулиці* INNER JOIN *Будівлі*

ON *Вулиці.Код_вулиці = Будівлі.Код_вулиці*;

Результатом запиту буде таблиця, в якій перелічені вулиці міста та номери будинків, які на них розташовані.

Питання для самодіагностики

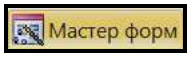
1. Для чого використовуються запити у БД?
2. Які види запитів ви знаєте? З якою метою вони використовуються?
3. Який із видів запиту потрібно використати для переміщення даних з однієї таблиці в іншу? Поясніть це на прикладі.
4. Опишіть способи створення запиту на вибірку.
5. Опишіть способи створення запитів з групуванням даних.
6. Опишіть способи створення перехресного запиту.
7. З якою метою використовуються перехресні запити?
8. У разі створення запитів звідки можна вибирати поля?
9. Яким чином при відображенні поля у запиті потрібно задати його характеристики?
10. Яким чином слід здійснювати у запитах MS Access сортування? З якою метою це використовується?
11. Які засоби MS Access можна використовувати для створення обчислюваного поля?
12. Які існують види з'єднань між таблицями в запитах? Як їх встановлювати?
13. Що потрібно зробити, щоб у вікні БД встановити режим відображення запитів?

3.3 Створення та редагування форм, звітів і кнопкових форм

Важливим моментом при завантажуванні даних у таблиці є забезпечення захисту від помилок. Тому потрібно дуже ретельно продумати питання, як забезпечити введення, редагування чи перегляд даних у вигляді, який був би зручним і звичним для користувача, а також запобігав би введенню помилкових даних. Таку можливість у MS Access можуть надати форми.

Форма в Access – це засіб вводу, відображення та редагування даних, розміщених у таблицях. Коли джерелом даних для завантаження в базу даних є первинний документ, то форма має відображати «формуляр-шаблон» цього об′єкта. Це полегшує занесення даних у таблиці, зменшує кількість можливих помилок, які могли б виникнути при введенні даних у таблиці. Форма є засобом організації інтерфейсу між користувачем та базою даних. Форми можна створювати на базі таблиць і запитів.

Створення форм за допомогою майстра

Для створення форми необхідно у вікні відкритої бази даних за допомогою кнопки 

Мастер форм панелі **Формы** вкладки **Создание** стрічки бази даних запустити майстер створення форм.

На першому кроці майстра слід вибрати необхідні таблиці чи запити та поля з них. Якщо ми створюємо форму з відомостями про будинок, то необхідно з таблиці **Будівлі** вибрати всі необхідні поля.

На другому кроці MS Access запропонує такі можливі варіанти побудови форм: **в один столбец**, **ленточную**, **табличную**, **выровненную**. Виберемо форму у вигляді стовпчика – всі поля будуть розміщені в один стовпчик.

На третьому кроці дамо назву майбутній формі, і якщо жодних установок більше робити не потрібно, тобто процес створення форми завершено, натиснемо кнопку **Готово**.

Зовнішній вигляд форми одразу після створення майстром наведено на рис. 3.34. Безумовно, він потребує редагування.

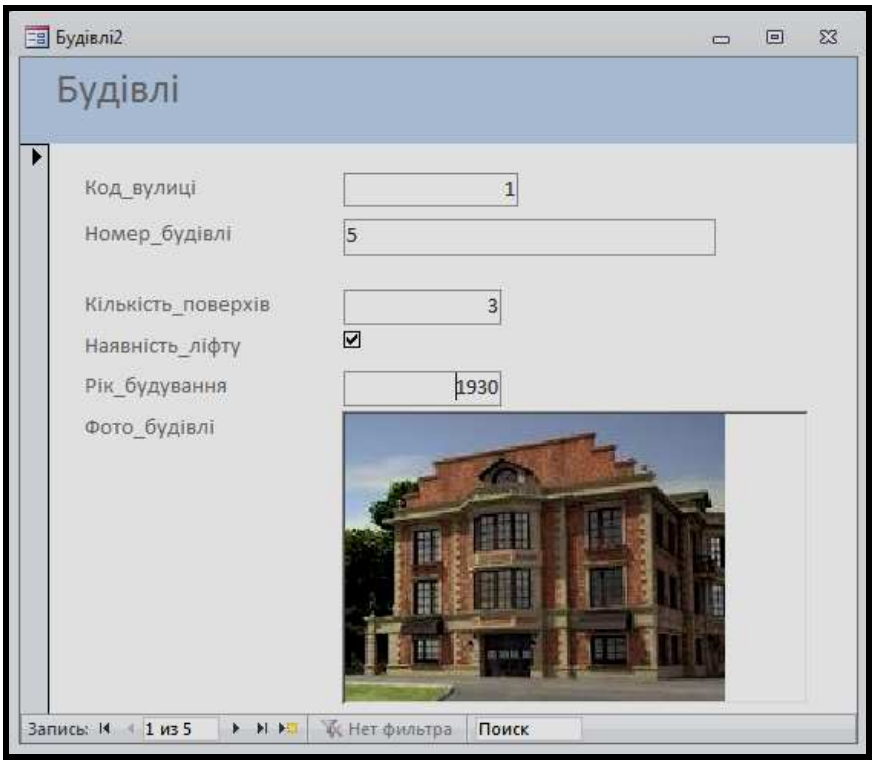


Рис. 3.34 – Вигляд форми після створення за допомогою майстра

Також існує можливість формування простої форми. Це є найбільш швидким способом створення. Його можна запустити на виконання за допомогою кнопки 

Форма панелі **Формы** вкладки **Создание** стрічки бази даних. Якщо вибрати цю піктограму, то система створить форму за один крок на основі вибраної таблиці чи запиту.

Створення форм на базі пов'язаних таблиць

В MS Access є можливість створити **комбіновану форму**, дані якої при її заповненні будуть заноситись до кількох таблиць. За допомогою таких форм можна виконувати не лише одночасне завантаження, а й перегляд і редагування даних пов'язаних між собою таблиць.

Форма на базі пов'язаних таблиць складається з головної та підпорядкованої форм, що є, так би мовити, формами всередині форми.

При створенні комбінованих форм необхідно дотримуватися таких умов:

- головна форма повинна базуватись на таблиці;
- підпорядкована форма може базуватись на таблиці, яка пов'язана з головною таблицею чи створена на базі таблиці або запиту, який вміщує поле з тим самим типом даних, що й ключове поле головної таблиці основної форми.

Створення комбінованої форми розглянемо на прикладі обліку квартир будівель, тобто всіх даних, включаючи відповідального квартиронаймача. Як головну форму використаємо вже створену форму *Будівлі*. Вона була створена на базі головної таблиці *Будівлі* і тому підійде для цього.

Створення додаткових елементів форми

Додаткові елементи форми створюють у режимі конструктора. Перейти в цей режим можна за допомогою кнопки 

Конструктор панелі **Режимы** вкладки **Главная** стрічки бази даних.

Управління безпомилковим уведенням даних

До елементів, що полегшують і забезпечують безпомилкове введення даних, належать такі засоби MS Access: прапорці та перемикачі, поля зі списками, встановлення початкового значення за замовчуванням, забезпечення обов'язкового введення даних у поле, блокування введення даних у деякі поля.

Створення полів зі списком

Розглянемо створення полів зі списком на прикладі створеної форми **Будівлі**. Визначення назви вулиці за кодом незручне і неінформативне. Замінимо його на поле зі списком, яке дозволить вибирати назву вулиці, в разі необхідності зміни даних, із випадаючого списку. Поля зі списком створюють у такій послідовності.

Переведіть форму в режим конструктора, переконайтеся що на панелі **Элементы управления** вкладки **Конструктор** активна кнопка  **Использовать мастера**. Виберіть команду поле зі списком  і клацніть мишкою в тому місті форми, де бажаєте побачити список.

Після цього на екрані з'явиться перше діалогове вікно процесу створення полів зі списком, який складається з кількох кроків. Після кожного з кроків необхідно натискати кнопку **Далее**.

У першому діалоговому вікні потрібно вибрати опцію, яка дозволить вибрати значення з другої таблиці чи запиту.

На другому кроці виконують вибір необхідної таблиці. У даному випадку вибирають таблицю-довідник *Вулиці*.

Після вибору таблиці з'явиться наступне вікно, в якому буде запропоновано вибрати необхідні поля з цієї таблиці. Вибравши поля *Код_вулиці* та *Назва_вулиці* натисніть кнопку **Далее**.

На наступному кроці з'явиться вікно, в якому буде запропоновано

здійснити сортування даних списку, якщо це необхідно.

На п’ятому кроці з'явиться вікно, в якому ви побачите вигляд списку, і буде запропоновано вибрати оптимальну ширину колонки для поля зі списком. Для цього необхідно двічі натиснути мишкою на правий край заголовка колонки.

Далі потрібно вказати поле на формі, в яке буде збережено створений список значень. У даному випадку це поле *Код_вулиці*.

На останньому кроці буде запропоновано задати ім’я створеному полю зі списком. Після натискання **Готово** автоматично збережеться ім'я, запропоноване MS Access.

Після додавання поля зі списком, яке показує назву вулиці, а не її код, блокування цього поля від змін і внесення деяких змін щодо оформлення форми зовнішній вигляд форми **Будівлі** буде таким як на рис. 3.35. Для того щоб запобігти внесенню змін у поля форми, їх можна заблокувати змінивши властивості. Для цього викликайте **Окно свойств** за допомогою кнопки 

Страница свойств, що на панелі **Сервис** вкладки **Конструктор**. На вкладинці **Все** цього вікна знайдіть властивість **Доступ**, виберіть значення **Нет**. Для властивості **Блокировка** виберіть значення **Да**. Також можна задати колір фону поля за допомогою властивості **Цвет фона**. У разі необхідності слід приховати поле на формі, тобто не виводи його на екрані монітора, залишивши на формі, і змінити властивість **Вывод на экран** на **Нет**.

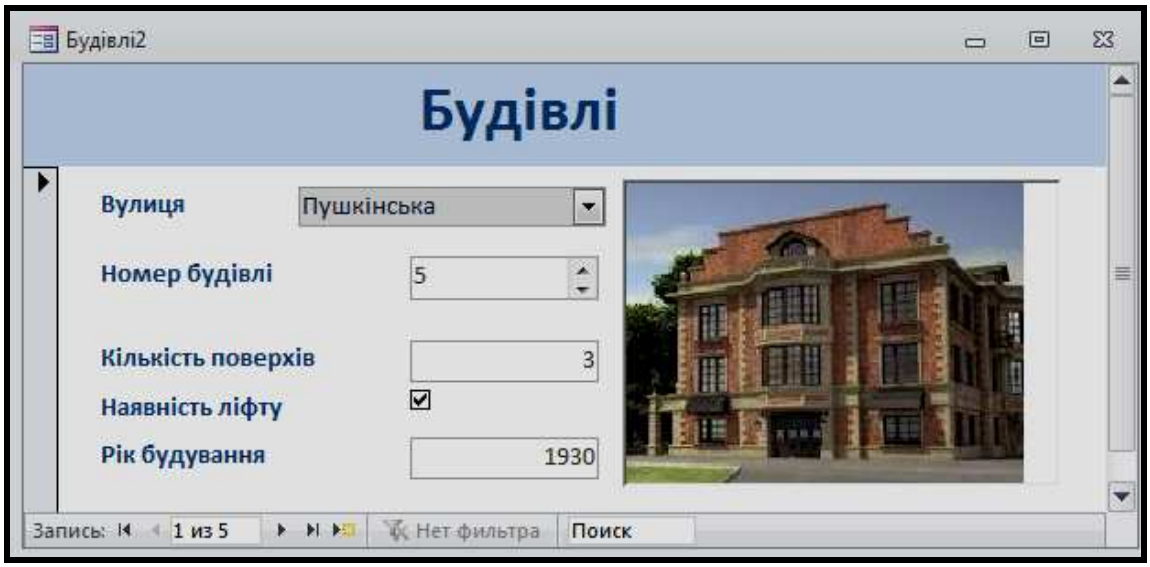


Рис. 3.35 – Остаточний вигляд головної форми

Підпорядковану форму побудуємо на базі запиту, створеного спеціально для підпорядкованої форми, в якому є всі потрібні для форми поля. Зовнішній вигляд форми необхідно вибрати стрічковий та обов′язково додати поле *Код_вулиці* для зв′язку головної і підпорядкованої форм.

Обчислювані елементи управління

Форма може вміщувати обчислювані вирази. Розглянемо приклад використання обчислюваного елемента управління на прикладі додавання у підпорядковану форму обчислюваного поля, яке буде показувати підсумкову загальну площу квартир, розташованих в одній будівлі.

Для цього підпорядковану форму **Квартири** необхідно відкрити у режимі конструктора, встановити покажчик миші на нижній межі області **Примечание формы** і розтягнути її вниз. Потім викликати елемент управління **Поле**  , що на панелі **Элементы управления** вкладки **Конструктор,** змінити текст надпису на **Підсумок**. У вікні властивостей цього поля на вкладці **Данные** у рядку **Данные** клацніть на кнопку виклику **построителя выражений**  . У вікні **построителя выражений** необхідно у списку **Элементы выражений** вибрати **Встроенные функции** з категорії **Статистические** і у стовпчику **Значения выражений** вибрати функцію **Sum**. Як аргумент слід підставити поле форми *Загальна площа*.

Імена полів у таких виразах беруть у квадратні дужки. Оскільки MS Access не може самостійно визначити тип поля, що обчислюється при створенні форми, то значення властивості **Формат поля** відповідного елемента необхідно замінити за допомогою **Окно свойств** на **С разделителями разрядов**, **Число десятичных знаков** – 2, також можна змінити колір тексту.

Роботу створеного елементу управління можна перевірити, викликавши **Режим формы** (рис. 3.36).

Для об′єднання підпорядкованої форми з головною необхідно відкрити головну форму в режимі конструктора та за допомогою команди  **Подчиненная форма/отчет**, що на панелі **Элементы управления** вкладки **Конструктор**, намалювати прямокутник у тому місті, де буде розташована підпорядкована форма. Далі включиться в роботу майстер підпорядкованих форм.

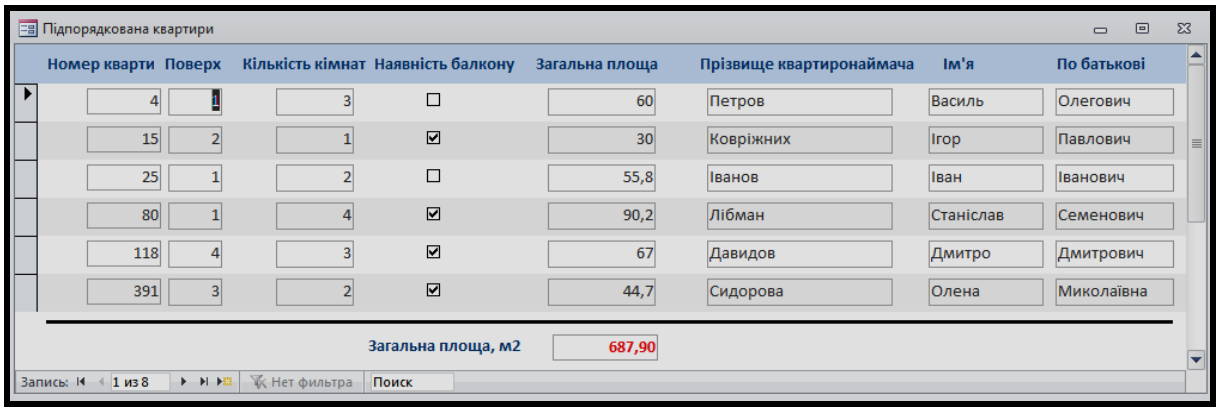


Рис. 3.36 – Підпорядкована форма з обчислюваним полем

У першому діалоговому вікні потрібно вибрати щойно створену підпорядковану форму.

У другому вікні майстра слід вказати поля зв′язків між формами, встановивши перемикач **Самостоятельное определение,** та вибрати поле **Код_будівлі**.

В останньому вікні майстра треба дати ім′я підпорядкованій формі та натиснути на кнопку **Готово**.

Головна форма утворена з полів таблиці **Будівлі**, а підпорядкована – з полів заздалегідь створеного запиту. Особливістю створеної підпорядкованої форми є те, що вона відображає лише ті записи таблиці **Квартири**, які розташовані на обраному записі таблиці **Будівлі** (рис. 3.37).

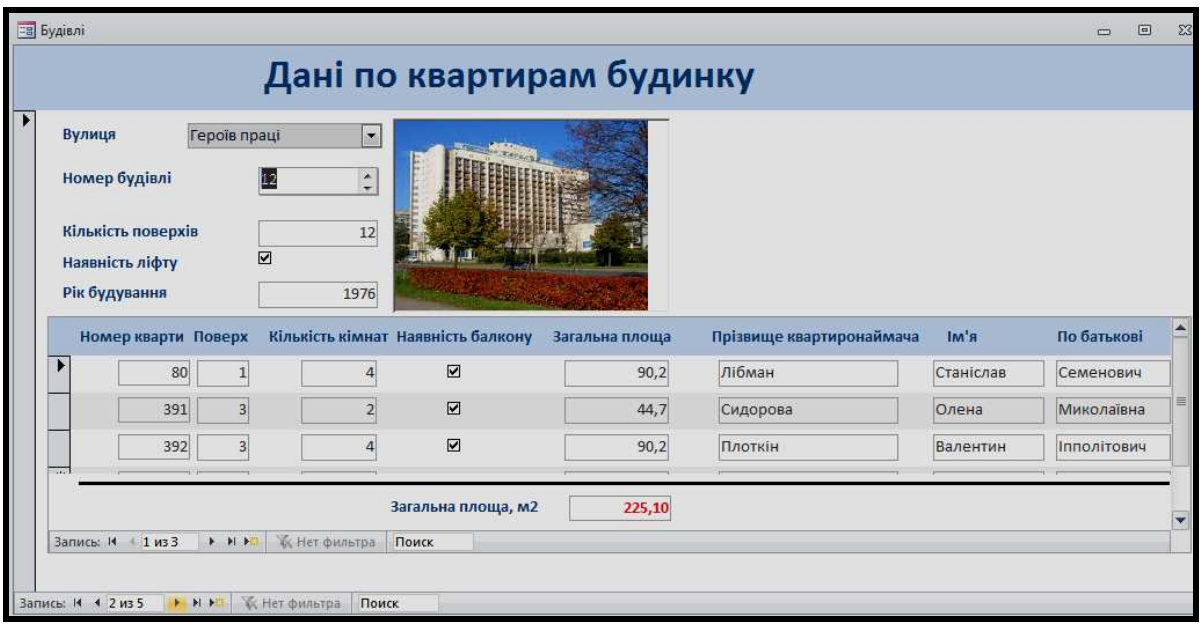


Рис. 3.37 − Зовнішній вигляд підпорядкованої форми

При введенні даних про новий будинок за допомогою форми

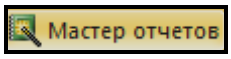
MS Access автоматично пов'язує записи з даними з таблиці **Квартири**, тому поле **Код_будинку** не потрібно буде ще раз заповнювати.

Створення звітів

Звіт є формою спеціального типу, призначеною для виведення підсумкової інформації на друк або на екран. Звіт може містити елементи управління і графічні компоненти. Отже, методи проектування звітів та форм у MS Access практично однакові.

Розрізняють прості звіти та звіти у вільній формі. Прості звіти створюються простими стандартними засобами і є роздруками таблиць чи звітів з даними, впорядкованими за рядками та стовпцями. Їх недолік – табульоване подання даних. Звіти у вільній формі створюються за допомогою спеціального конструктора, в них поля початкової таблиці розташовуються по вертикалі або у відведених для них місцях. Відсутність обмежень на розміщення полів значно розширює область застосування таких звітів.

Створення будь-яких звітів проводиться таким чином – відкривається вікно вибраної бази даних і активізується вкладка **Создание,** де на панелі **Конструктор** обирається потрібний вид звіту.

Команда **Мастер отчетов** 

на панелі **Отчеты** дозволяє створювати складніші звіти на основі однієї або декількох пов'язаних таблиць бази даних. Він здатний групувати і сортувати записи, формувати підсумкові поля, задавати вид макету звіту та стиль його оформлення, а також переглядати й модифікувати звіти.

Конструктор – це наймогутніший засіб проектування складних звітів. З його допомогою можна створювати звіти з будь-якою конфігурацією і оформленням, а також упроваджувати у звіт будь-які елементи управління і графічні об'єкти. Його результат – друкований звіт у вільній формі.

Звіт у вигляді **Наклейки** реалізується відповідним майстром. При цьому на основі даних таблиць чи звітів створюються поштові наклейки з різними іменами, зовнішнім виглядом і розмірами.

Створення звітів за допомогою майстра

Майстер звітів дозволяє створювати різноманітні за стилем оформлення звіти, що містять поля однієї або декількох таблиць. Можна також створювати звіти і на основі запитів. Розглянемо створення звіту за допомогою майстра на прикладі створення списків мешканців по квартирах.

На першому кроці роботи майстра відкривається вікно **Создание**

отчетов, де вибираються початкові таблиці або запити і їх поля. Методика вибору полів для звіту залишається такою ж, як і для запитів та форм.

На другому кроці роботи майстра проводиться групування даних за обраним полем, наприклад, за полем *Назва_вулиці*. Звіт може мати до чотирьох рівнів групування, які призначаються користувачем або майстром за замовчуванням. Додавання (видалення) рівнів групування, а також підвищення (пониження) рівня групування полів виконується за допомогою кнопок  вікна майстра. При натисканні на кнопку **Группировка** відкривається додаткове вікно діалогу, де вибирається інтервал групування. При використанні інтервалу **Обычный** в одну групу об'єднуються записи з однаковими значеннями вибраного поля. Для числових полів можна також задавати групування за десятками, сотнями (розрядністю), а для текстових полів – за першою буквою або декількома першими буквами.

На третьому кроці роботи майстра задається порядок сортування записів звіту, а також тип підсумкових операцій над числовими полями. Таким чином, майстер забезпечує сортування записів звіту в порядку убування (зростання) значень від одного до чотирьох полів, вибір підсумкових полів та відповідних функцій Sum, Avg, Min і Max.

На четвертому кроці роботи майстра вибирається вид макету звіту й орієнтація сторінки звіту.

На п′ятому кроці роботи майстра обирається макет звіту – **Ступенчатый**, **Блок** чи **Структура**, а також орієнтація сторінки звіту – **Книжная** чи **Альбомная**.

На завершальному кроці створення звіту задається ім'я звіту і вибирається варіант подальших дій користувача – можна переглянути звіт або змінити макет звіту.

Після перегляду зовнішнього вигляду звіту відкриємо його в режимі конструктора з метою модифікації. Конструктор дозволяє видаляти, додавати і переміщати об'єкти звіту, вбудовувати у звіти елементи управління та графічні об'єкти, а також виконувати безліч процедур з поліпшення зовнішнього оформлення звіту.

Конструктор звітів і його застосування

Відомо, що конструктор звітів дозволяє користувачеві створювати складні звіти із заданими параметрами, які містять елементи управління і графічні об'єкти. При цьому процес створення звітів багато в чому збігаєть-

ся з процесом створення форм, хоча й має ряд специфічних особливостей. Вікно конструктора звітів відкривається за допомогою команди **Конструктор** 

, що на панелі **Режимы** вкладки **Конструктор**. Вікно конструктора, як і вікно конструктора форм, може містити: координатну сітку, масштабні лінійки, область даних, а також області заголовку, примітки, верхній і нижній колонтитули. Заголовок і примітка звіту включаються чи вимикаються за допомогою команди **Заголовок/Примечания отчета** з контекстного меню, а верхній і нижній колонтитули – за допомогою команди **Колонтитулы страницы**. Що стосується області даних, то вона включається системою за замовчуванням і міститься у вікні конструктора завжди.

Для додавання у звіт елементів управління використовують команди панелі **Элементы управления** вкладки **Конструктор**. Попередній перегляд і друк звітів у режимі конструктора проводиться у звичайному порядку. За наявності у звіті груп записів цей порядок практично зберігається, але спочатку і в кінці кожної групи поміщаються відповідні заголовки та примітки. Для кожної групи можна також створити й області верхнього і нижнього колонтитулів.

До специфічних процедур для роботи з конструктором звіту відносяться: введення дати, друк звіту, нумерація сторінок і записів звіту, групування і сортування записів, відображення проміжних сум, розміщення даних у декількох колонках, створення поштових марок.

Нумерація записів звіту

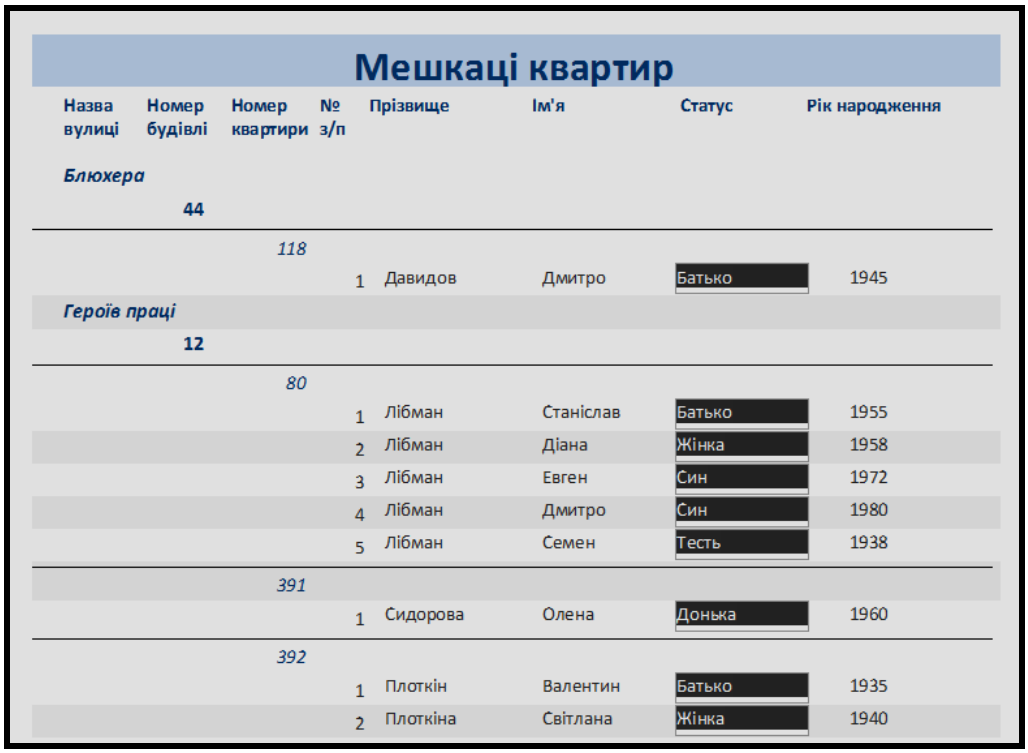
У MS Access можуть нумеруватися не тільки сторінки, але і їх записи. Причому нумерація записів можлива як по всьому звіту, так і в межах груп. Для цього необхідно:

ввести в область даних конструктора незв'язане поле, тобто поле без напису;

відкрити вікно властивостей поля і на його вкладинці **Данные** у рядку **Данные** вписати значення *=1;*

у рядку **Сумма с накоплением** вписати значення **Для всего** або **Для группы**.

У результаті записи звіту будуть пронумеровані (рис. 3.38).



Мешкаці квартир

Назва вулиці	Номер будівлі	Номер квартири	№ з/п	Прізвище	Ім'я	Статус	Рік народження
Блюхера							
	44						
		118					
			1	Давидов	Дмитро	Батько	1945
Героїв праці							
	12						
		80					
			1	Лібман	Станіслав	Батько	1955
			2	Лібман	Діана	Жінка	1958
			3	Лібман	Евген	Син	1972
			4	Лібман	Дмитро	Син	1980
			5	Лібман	Семен	Тесть	1938
		391					
			1	Сидорова	Олена	Донька	1960
		392					
			1	Плоткін	Валентин	Батько	1935
			2	Плоткіна	Світлана	Жінка	1940

Рис. 3.38 – Звіт із пронумерованими записами

Створення обчислюваних полів у звітах

Створення обчислюваних полів у звітах виконується в такому порядку:

- за допомогою команди **Поле**  , що на панелі **Элементы управления** вкладки **Конструктор**, у необхідну область звіту вводиться пов'язане поле і відкривається вікно його властивостей із вкладкою **Данные**;
- клацанням на кнопці  у рядку **Данные** активізується **Построитель выражений**, засобами якого формується розрахункова формула;
- за допомогою кнопки **ОК** формула спочатку передається в рядок властивості **Данные**, а потім – у виділене поле конструктора;
- на закінчення в поле напису вводиться ім'я обчислюваного поля.

Вставка графічних об'єктів

Порядок вставки графічних об'єктів у звіти залишається таким, як і у формах:

- відкрити вікно конструктора звіту і виконати команду **Рисунок**  , що на панелі **Элементы управления** вкладки **Конструктор**. Вставити рисунок в область заголовка (рис. 3.39);

- встановити потрібні розміри рамки картинки шляхом буксирування її граничних маркерів. Відкрити вікно властивостей картинки і на вкладці **Макет** у рядку **Установка размеров** вибрати слушний варіант розташування;
- перейти в режим попереднього перегляду й оцінити зовнішній вигляд документа.

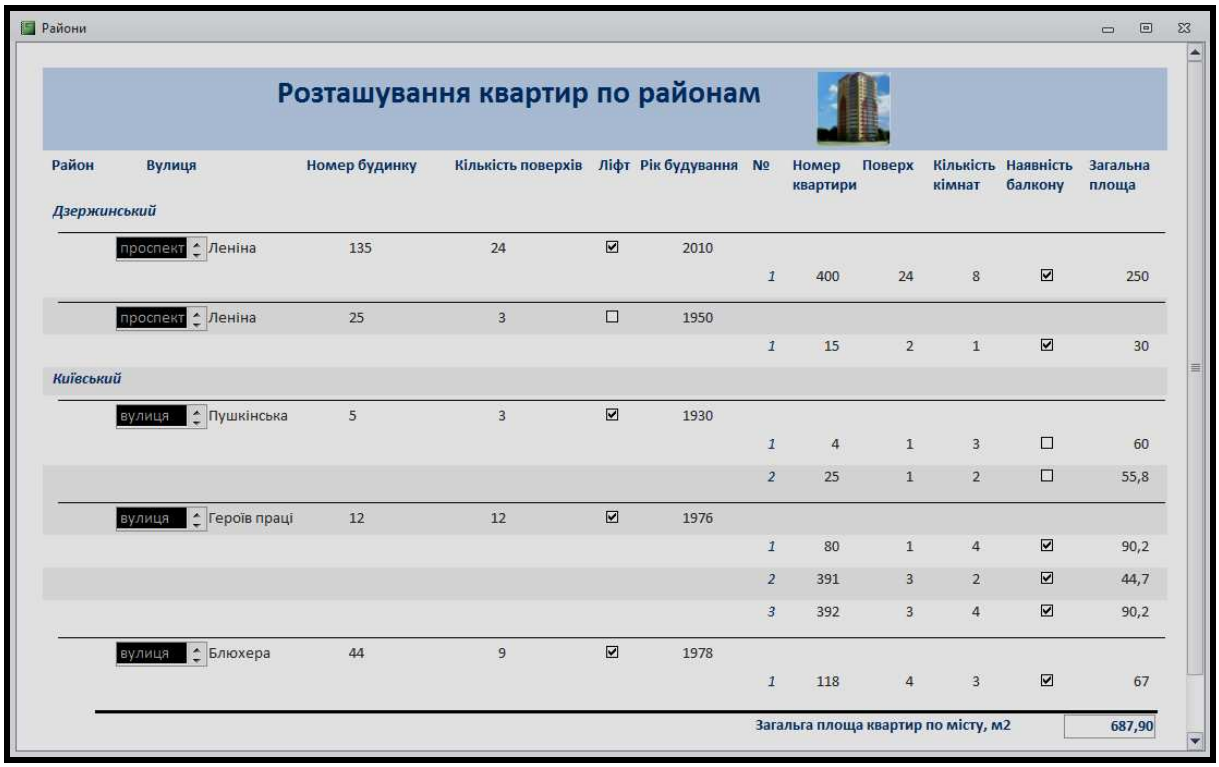
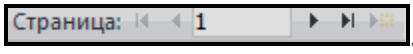


Рис. 3.39 – Багаторівневий звіт із підсумками

Перегляд і друк звітів

Вище наголошувалося, що будь-який звіт можна проглянути на екрані монітора за допомогою команди **Предварительный просмотр** 

, що на панелі **Режимы** вкладки **Конструктор**, або за допомогою контекстного меню виділеного звіту.

Для проглядання сторінок звіту використовуються кнопки панелі навігації 

.

Друк поточного звіту виконується за допомогою команди 

Печать, що на панелі **Печать** вкладки **Предварительный просмотр**. Параметрами друку можна встановити такі:

- тип використовуваного принтера;

- номери сторінок, що підлягають роздрукуванню;
- число друкованих копій звіту.

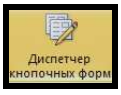
Після установки параметрів друку і підготовки принтера до роботи звіт роздруковується.

Створення додатків із використанням макросів і мови VBA

Створення головної кнопкової форми

Проектування схеми керування створеної бази даних (застосування) виконується за технологією «зверху-донизу», тобто від головної кнопкової форми до функціональних форм і звітів, що викликаються. Оскільки всі функціональні форми і звіти вже існують, то залишилося побудувати тільки кнопкові форми (рис. 3.40). Їхнє створення будемо виконувати в такому порядку:

- створення кнопкової форми *Довідники*;
- створення кнопкової форми *Облік*;
- створення кнопкової форми *Аналіз*;
- створення *головної кнопкової форми*;
- усунення недоліків.

При створенні кнопкової форми будемо користуватися майстром. У разі відсутності на стрічці робочого простору MS Access кнопки **Диспетчер кнопочных форм** 

налаштуємо його. Для цього необхідно на вкладці **Файл** вибрати команду **Параметры**. У вікні **Параметры Access**, що з′явилося вибрати розділ **Настройка ленты**. Далі необхідно на будь-якій вкладці за допомогою контекстного меню створити нову групу і добавити на неї команду **Диспетчер кнопочных форм**. Запуск майстра виконується клацанням по щойно добавленій кнопці.

Створення кнопкової форми «Довідники»

Створимо кнопкову форму ***Довідники*** відповідно до макета (рис. 3.41).

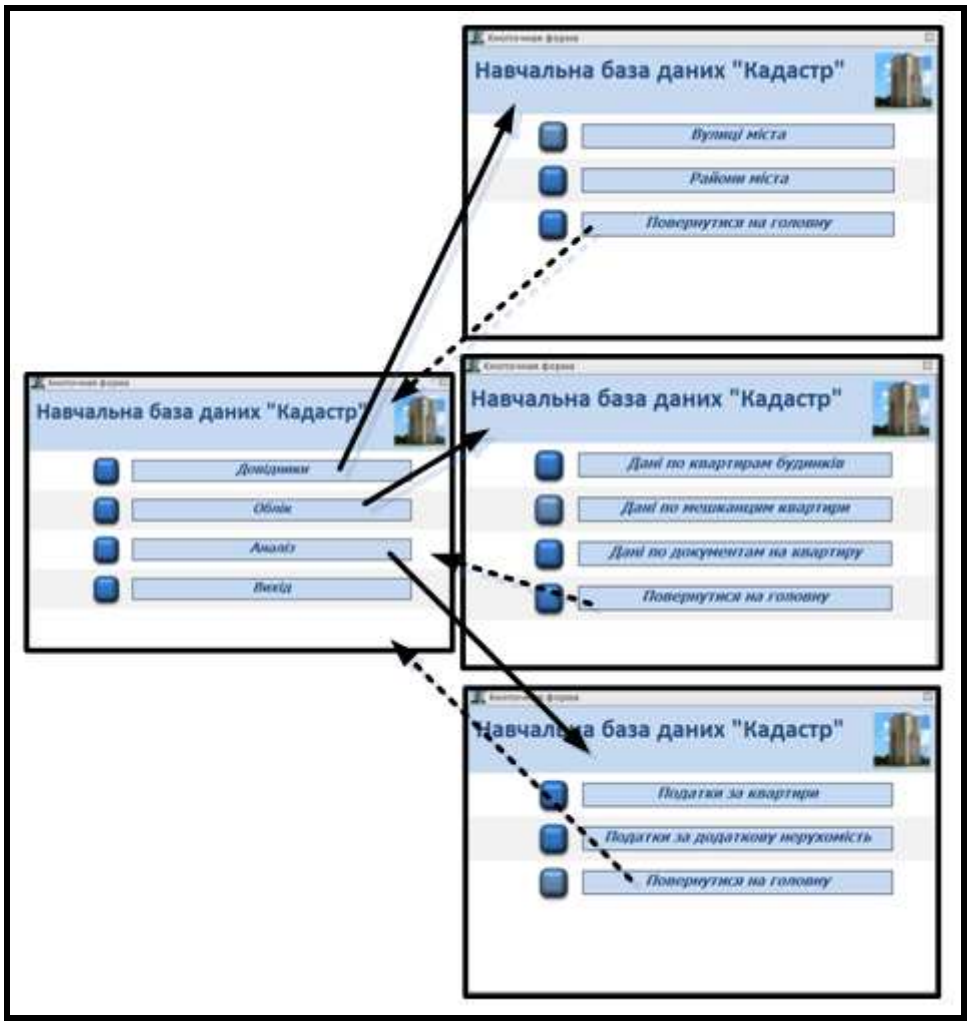


Рис. 3.40 – Схема керування застосуванням за допомогою кнопкової форми

Рис. 3.41 – Макет форм «Довідники»

Створення кнопкової форми виконується в такому порядку:

1. Клацніть на кнопці **Диспетчер кнопочных форм**. У відповідь на запит, чи створити кнопкову форму, клацніть на кнопці **ДА**.

2. У вікні **Диспетчер кнопочных форм**, що з′явилося, клацніть на кнопці **Создать**.

3. У вікні **Создание** введіть ім′я нової сторінки кнопкової форми *Довідники* і клацніть на кнопці **ОК**.

4. Після повернення у вікно **Диспетчер кнопочных форм** клацніть на елементі *Довідники*, а потім на кнопці **Изменить**, щоб додати кнопки на кнопкову форму.

5. У вікні **Изменение кнопочной формы** клацніть на кнопці Создать, щоб описати нову кнопку *Райони міста*.

6. У вікні **Изменение кнопочной формы** потрібно встановити такі значення нової кнопки:

- у полі **Текст** увести назву кнопки *Райони міста*;
- у полі **Команда** виберіть дію **Открыть форму для добавления**;
- у полі **Форма** виберіть назву форми *Райони міста* і клацніть на кнопці **ОК**;
- клацніть на кнопці **Закрыть**, щоб повернутися у вікно **Диспетчер кнопочных форм**.

Створення кнопкової форми *Облік* і додавання їх на головну кнопкову форму виконується за аналогією.

Усунення недоліків

Переведіть головну кнопкову форму в режим конструктора і викличте її властивості за допомогою подвійного клацання мишею на перетині горизонтальної та вертикальної лінійок. У вікні **Окно свойств**, що з′явилося, перейдіть на вкладку **Все** і задайте такі значення властивостей:

- **кнопка оконного меню** – Нет;
- **кнопка закрытия** – Нет;
- **тип границы** – отсутствует.

Налаштування параметрів автозапуску

У разі необхідності налаштуйте застосування ***Кадастр*** таким чином, щоб користувач працював із даними бази тільки через кнопкові форми. Для цього потрібно налашьувати параметри автозапуску MS Access. Для того щоб після запуску застосування відразу відображалася головна кноп-

кова форма, а вікно бази даних було відсутнє і у заголовках всіх вікон застосування було видно відповідний значок, потрібно виконати такі дії:

1. На вкладниці **Файл** вибрати команду **Параметры**. У вікні **Параметры Access**, що з′явилося, вибрати розділ **Текущая база данных**.

2. У розділі **Параметры приложений** вибрати головну кнопкову форму як **форму просмотра,** а також за допомогою кнопки **Обзор** вказати розташування малюнка застосування.

3. У розділі **Навигация** зняти прапорець із команди **Область навигации**.

4. У розділі **Навигация** зняти прапорець із команди **Параметры ленты и панелей инструментов**.

Після цього слід закрити базу даних. При подальшій роботі з базою даних *Кадастр* користувач зможе виконувати навігацію тільки за допомогою кнопкової форми, тобто буде тільки переглядати та додавати в разі необхідності нові дані, вносити зміни у структуру бази даних він не зможе.

У разі необхідності повернутися до режиму редагування створеного додатка треба в момент відкриття бази даних, тримаючи натиснутою клавішу **Shift**.

Створення кнопки на формі за допомогою макросу

На сторінці *Облік* головної кнопкової форми створимо перехід до форми *Дані по квартирах будинку* за допомогою макросу.

Створення макросу виконується в такому порядку:

- На вкладці **Создание** панелі **Макросы и код** натисніть кнопку **Макрос** 

.
- Натисніть кнопку **Сохранить** на панелі швидкого доступу, введіть ім’я макросу *ДоФормиКвартири* у вікно, що з'явилося, і натисніть кнопку **ОК**.
- У вікні, що з′явилося, виберіть елемент **Открыть Форму** у списку, що розкривається.
- У полі **Имя формы** виберіть ім′я форми ***Будівлі***.
- У полі **Режим данных** виберіть у списку, що розкривається, **Изменение**.
- У полі **Режим окна** залиште **Обычное**.
- Закрийте вікно конструктора макросів із збереженням зроблених змін (рис. 3.42).

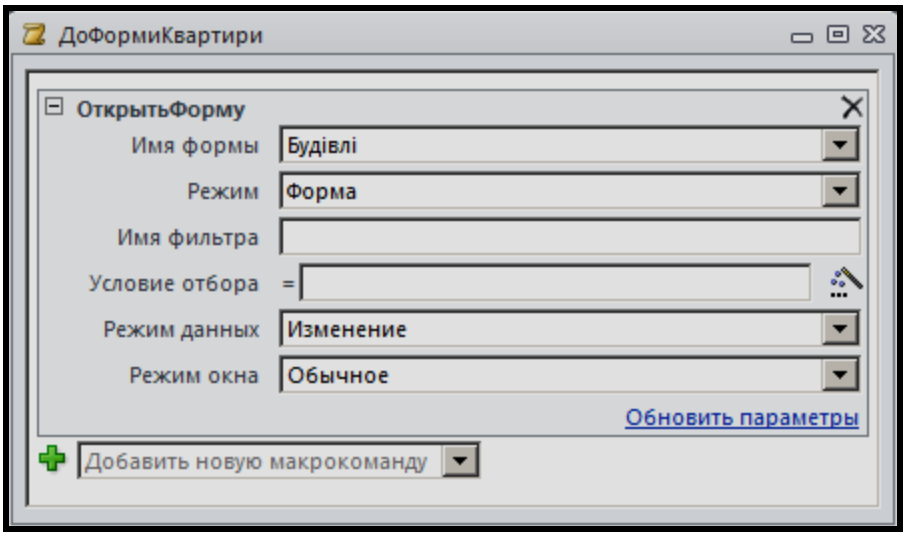


Рис. 3.42 – Формування макросу

Створений таким чином макрос не виконує ніяких дій, поки не станеться подія, з якою він пов’язаний, тобто необхідно створити зв′язок макросу з кнопкою або текстом на сторінці *Облік* головної кнопкової форми. Для цього необхідно виконати таку послідовність дій:

1. Відкрийте форму *Облік* у режимі конструктора.
2. За допомогою вкладки **Конструктор** панелі **Элементы управления** створіть кнопку. У вікні, що з′явилося, майстра **Создание кнопок** натисніть **Отмена**.
3. Виділіть на формі кнопку і викличте вікно її властивостей.
4. У вкладці **События** клацніть у рядку **Нажатие кнопки**.
5. У списку, що розкривається, виберіть щойно створений макрос *ДоФормиКвартири* (рис. 3.43).
6. Збережіть форму й перейдіть у режим форми.

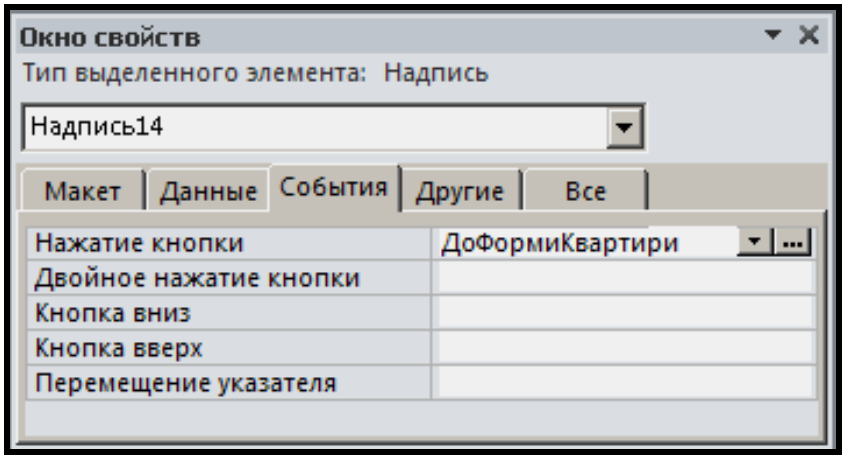


Рис. 3.43 – Оброблення події «натискання кнопки макросу»

Таким чином, за допомогою макросів можна додати всі необхідні кнопки на сторінки головної кнопкової форми, послідовно виконуючи описані вище дії.

Публікація баз даних у комп'ютерних мережах

Створення доступу до даних SharePoint за допомогою форм MS Access

Спочатку треба встановити на свій комп'ютер сервер із програмою SharePoint. Далі у програмі MS Access на вкладці **Файл** нажміть кнопку **Сохранить и опубликовать**, а потім – **Опубликовать в Access Services**. Далі для коректної публікації бази даних необхідно перевірити її на помилки (у Web-бази даних є деякі обмеження порівнянно з персональною). Для цього натисніть кнопку **Проверка совместимости** 

. Перевірка забезпечує коректну публікацію бази даних. У разі виявлення проблем їх необхідно вирішити до публікації.

Примітка. Інформація про виявлені проблеми зберігається в таблиці під заголовком «**Ошибки веб-совместимости**». Кожний рядок у таблиці містить посилання на відомості про усунення недоліків.

У розділі **Опубликовать в Access Services** введіть URL-адресу сервера SharePoint, на якому буде опублікована база даних, та вкажіть ім'я веб-бази даних. Це ім'я разом із URL-адресою сервера утворює URL-адресу додатка (рис. 3.44).

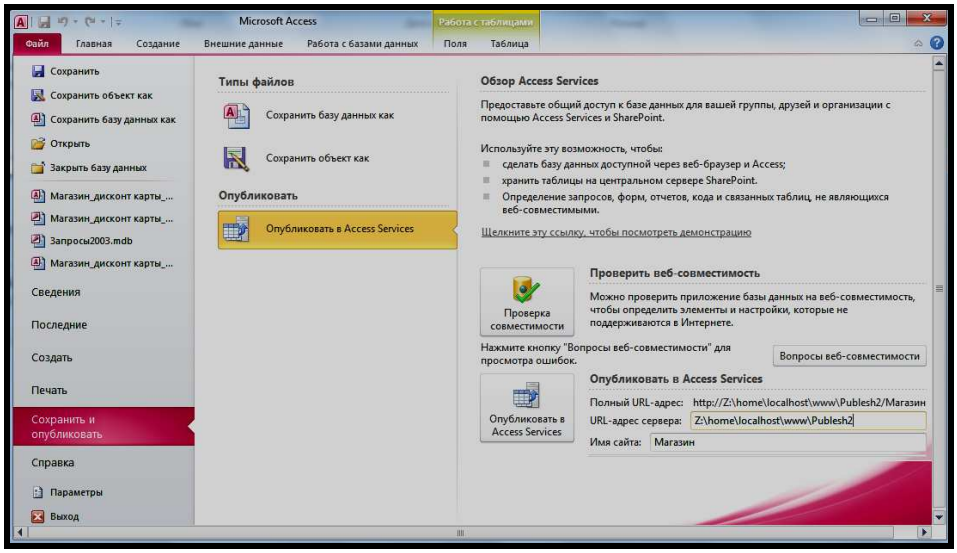


Рис. 3.44 – Публікація бази даних на сервері SharePoint

У результаті база даних буде розташована на сервері, а користувачі

одержать за допомогою програм перегляду і наявності доступу до мережі Internet/Intranet форми для роботи з Web-базою даних «*Кадастр*».

Необхідно підкреслити, що стрімке розповсюдження мережі Internet, поява системи WWW, створення Web-серверів і програм перегляду Web-сторінок, що містять засоби роботи з формами, фактично усунуло перешкоду для супроводу баз даних. На даний час завдяки використанню Web-сервера (як буфера між базою даних і мережею) та програми перегляду Web (як зовнішнього інтерфейсу, незалежного від операційної системи, встановленої на комп′ютері) практично кожний користувач одержує можливість організації мережевої публікації бази даних, причому із забезпеченням належної якості та конфіденційності інформації. Такі проблеми, як несумісність старих додатків або різних операційних систем, платне ліцензійне програмне забезпечення (бази даних), тепер мають значно меншу гостроту або взагалі зникли завдяки доступності Web-серверів і програм перегляду (браузерів).

Питання для самодіагностики

1. Для чого призначені форми у базі даних?
2. Як створюються додаткові елементи форми?
3. Як створити на формі обчислюване поле?
4. Які види форм ви знаєте? Охарактеризуйте кожен з них.
5. У якій області форми розташовують підсумкові значення?
6. Які розділи звіту ви знаєте? Для відображення яких даних вони використовуються?
7. На основі яких об′єктів бази даних можуть будуватися звіти?
8. Яким чином пронумерувати рядки у звіті?
9. Які види звітів ви знаєте? Охарактеризуйте кожен з них.
10. Скільки рівнів групування можна встановлювати у звітах MS Access?
11. У якому режимі виконують редагування звітів? Як у нього перейти?
12. Які функції роботи з даними можна додати на сторінку доступу?
13. Яким чином організовується безпека доступу до даних?
14. Де повинні розташовуватися файли сторінок доступу?
15. Що становить схема керування застосуванням?
16. З якою метою використовують кнопкові форми?

РОЗДІЛ 4
ЗАВДАННЯ НА ЛАБОРАТОРНІ ЗАНЯТТЯ ТА ІНДИВІДУАЛЬНЕ НАВЧАЛЬНО-ДОСЛІДНЕ ЗАВДАННЯ

У даному розділі розглядаються: кроки виконання індивідуального навчально-дослідного завдання студента. Подані варіанти завдань, основні вимоги до створюваних баз даних та наведено приклад виконання індивідуального завдання.

4.1 Загальні вимоги до розроблюваної БД

1. Розроблювана БД повинна включати такі об’єкти: таблиці, запити, форми, звіти та макроси.

2. Схема бази даних, необхідна кількість таблиць, їх структура та властивості полів визначаються студентом самостійно, виходячи із постановки задачі згідно з варіантом завдання. Кількість записів у довідкових таблицях повинна бути не менше 7 – 10, а в таблицях із змінними даними – 30 записів.

3. Відбір даних для аналізу реалізується в запитах. Необхідна кількість запитів визначається студентом самостійно згідно з постановкою задачі, але в розроблюваній БД повинні бути подані всі види запитів: запити на вибірку даних з різними умовами відбору даних; запити з обчислюваними полями; підсумкові запити; перехресні запити; запити на оновлення, видалення та додавання даних, а також запити на створення таблиць.

4. Інтерфейс користувача реалізується за допомогою форм, кількість і зовнішній вигляд яких визначаються студентом самостійно, але в розроблюваній БД повинні бути створені такі форми: головна кнопкова форма; форми для редагування довідників; форма для редагування таблиць із змінною інформацією; форми з аналітичними даними, реалізовані у вигляді підпорядкованих або пов’язаних форм, які вміщують різні види діаграм. У БД обов’язково потрібно використовувати зведені таблиці і діаграми для аналізу процесів. Робота форм повинна бути автоматизована за допомогою макросів, які включають макрокоманди для роботи із записами, керування застосування та фільтри.

5. Кількість і зміст звітів визначається студентом самостійно відповідно до поставленої задачі, але у створених звітах повинно бути використано всі можливі розділи для звіту; реалізовано декілька (не менше двох)

рівнів групування даних; організована нумерація записів за допомогою поля із сумою з накопиченням; використані обчислювані поля.

Індивідуальні навчально-дослідні завдання (далі – ІНДЗ) мають на меті:

- систематизацію, закріплення, розширення теоретичних і практичних знань, умінь з питань створення проектів застосувань за допомогою СКБД Access;
- використання набутих знань і вмінь під час реалізації конкретних економічних проектів;
- набуття компетентності в галузі застосування технологій баз даних до розв'язання інформаційних задач у фаховій діяльності;
- розвиток навичок самостійної організації роботи при виконанні ІНДЗ.

Актуальність даного виду роботи зумовлена розповсюдженістю проблем збирання, зберігання та ефективної обробки інформації при організації та повсякденному веденні справ.

Отже, індивідуальні завдання є важливою частиною навчального модуля дисципліни і виконуються студентами самостійно під керівництвом викладачів, які забезпечують викладання дисципліни.

Теми ІНДЗ охоплюють широке коло різноманітних прикладних задач, які відповідають реальним ситуаціям і пов'язані єдиними підходами та методами для їхнього вирішення в середовищі MS Access: розробка логічної структури розподілу даних і зв'язків між ними, визначення відповідних типів та форматів для зберігання даних, створення дружнього інтерфейсу користувача, виконання необхідного аналізу даних, подання результатів аналізу у вигляді звітів тощо. Перелік тем ІНДЗ наведено в табл. 4.1.

Основною **метою** індивідуального завдання є формування у студентів основ професійного підходу до розв'язання інформаційних задач у майбутній фаховій діяльності: вміння сформулювати постановку задачі; знаходити, обмежувати та обґрунтовувати саме ті атрибути предметної області, які необхідні для її розв'язання; розуміння і вміння формулювати задачі аналізу, а також здійснювати програмну реалізацію з використанням системи куревання базами даних MS Access та інших застосувань MS Office.

Усі студенти виконують роботу за таким загальним змістом:

- постановка задачі. Розробка логічної схеми БД та перелік аналі-

тичних задач:

- створення таблиць БД, заповнення таблиць економічно обґрунтованими даними;
- проектування запитів як основного інструменту розв’язання інформаційних задач;
- проектування форм як основних елементів інтерфейсу користувача створюваного додатка (бази даних);
- технологія створення, редагування та використання звітів у базі даних MS Access;
- автоматизація та створення застосувань у БД.

Таблиця 4.1 – Шкала оцінювання ІНДЗ

Сума балів за всі види навчальної діяльності	**Оцінка ECTS**	**Оцінка за національною шкалою**	
		для екзамену, курсового проекту, ІНДЗ	**для заліку**
90 – 100	A	відмінно	зараховано
82 – 89	B	добре	
74 – 81	C		
64 – 73	D	задовільно	
60 – 63	E		
35 – 59	FX	незадовільно з можливістю повторного складання	не зараховано з можливістю повторного складання
0 – 34	F	незадовільно з обв′язковим повторним вивченням дисципліни	не зараховано з обв′язковим повторним вивченням дисципліни

Аналітична частина завдання полягає в постановці задачі та визначенні завдяки цьому необхідної для зберігання в базі інформації, побудові логічної схеми бази даних у вигляді пов’язаних між собою таблиць, визначенні властивостей атрибутів (полів) таблиць та властивостей зв’язків між ними.

Дослідницька частина роботи полягає в тому, що студенти повинні самостійно з’ясувати необхідну інформацію стосовно заданої предметної області, виконати розподіл інформації на довідкову і змінну, визначити такі обмеження щодо конкретних даних, які будуть зберігатися в базі, щоб ці дані відображали можливу реальну ситуацію в предметній області, а також повинні самостійно визначити коло аналітичних задач, пов’язаних із діяльністю, яка фіксується в базі.

Проектна частина ІНДЗ передбачає безпосередню роботу в MS Access і створення необхідних об’єктів бази із визначеними властивостями.

У результаті виконання індивідуального завдання студент повинен розробити інтерфейс користувача за допомогою відповідних форм та провести економічний аналіз даних за допомогою відповідних об’єктів бази.

Кожен студент оформляє звіти засобами MS Word, які зберігаються на будь-яких зовнішніх носіях інформації, роздруковуються та захищаються. Результати роботи та захисту оцінюються згідно зі шкалою, що подана в табл. 4.1.

4.2 Варіанти завдань

Для свого варіанта згідно зі студентським журналом (табл. 4. 2) розробити базу даних і створити файл бази даних у MS Access.

Таблиця 4.2 – Варіанти завдань

№ варианту	Назва	Загальна постановка задачі
1	2	3
1	Автомайстерня	Облік виконаних ремонтних робіт в автомайстерні для поточного та підсумкового аналізу роботи майстерні. Ведеться облік клієнтів та автомобілів
2	Мережа АЗС	Облік операцій із продажу різних видів палива в мережі АЗС. Ведеться облік поставок і постачальників
3	Аптека	Облік операцій із реалізації медичних препаратів при роздрібній торгівлі в окремій аптеці міста. Ведеться облік поставок препаратів і постачальників
4	Бібліотека	Облік операцій обслуговування читачів. Ведеться облік читачів, літератури, а також роботи персоналу бібліотеки
5	ДАІ. Технічний догляд автомобілів	Облік технічного стану автомобілів на рівні районної ДАІ
6	Клієнти готелю	Оперативний облік обслуговування клієнтів готелю. Ведеться облік клієнтів, персоналу, номерів
7	Деканат. Успішність студентів на одному курсі	Оперативний облік успішності студентів за дисциплінами одного курсу

Продовження табл. 4.2

1	2	3
8	Кадрова агенція	Облік резюме здобувачів та заявок роботодавців, а також облік виконаного підбору персоналу (укладених контрактів)
9	Магазин «Книжковий світ»	Облік роздрібного продажу книжок у книжковому магазині. Необхідно мати інформацію про постачальників і поставки літератури різних жанрів
10	Меблі на замовлення	Облік видів меблів, замовників, майстрів, матеріалів, комплектуючих
11	Обмін валют	Облік роздрібного продажу і купівлі валют у мережі пунктів обміну. У базі фіксується кожна проведена операція з обміну валюти, а також поточний курс купівлі та продажу
12	Продаж авіаквитків	Облік рейсів, пасажирів, продажу квитків авіакомпанією
13	Продаж залізничних квитків	Облік залізничних маршрутів, зон, продажу квитків
14	Розклад занять у ВНЗ	Облік аудиторій, дисциплін, викладачів, груп, розкладу
15	Склад. Облік матеріалів	Облік постачальників, споживачів, матеріалів, облік прийому/відпуску матеріалів
16	Ремонтна майстерня	Облік видів техніки, клієнтів, облік ремонтних операцій
17	Тур-фірма	Опис турів, клієнтів, продажу путівок
18	Фірма з продажу комп’ютерів	Облік комплектуючих ПК, формування конфігурації
19	Поліклініка	Оперативний облік звертань пацієнтів до різних лікарів, діагнози
20	Концерти	Організація концертів, артисти, номери, місця проведення
21	Спортивні досягнення	Облік змагань, суперників, результатів, складу команд
22	Виробництво продукції	Облік видів продукції, матеріалів і комплектуючих, собівартості продукції (планової і фактичної). Випуск продукції
23	Будівництво котеджу	Облік усіх заходів, трудових та матеріальних ресурсів, необхідних для будівництва невеличкої одно або двоповерхової будівлі

Закінчення табл. 4.2

1	2	3
24	Облік маршрутів міського транспорту	Облік маршрутів міського транспорту (автобусів, електротранспорту) з урахуванням кількості пасажирів та зупинок, а також графік руху
25	Паспортизація геодезичного устаткування	Облік термінів перевірок придатності та надійності роботи обладнання
26	Облік комунальних платежів мешканцями квартир	Облік щомісячної плати за квартиру, опалення, воду, каналізацію, електроенергію, газ

4.3 Виконання індивідуального навчально-дослідного завдання

Крок 1. Постановка задачі. Розробка логічної схеми БД та переліку аналітичних задач

Відповідно до індивідуального завдання описати конкретну предметну область, яка вимагає розроблення відповідної бази даних. Сформулювати перелік основних задач, які будуть розв'язуватися на основі даних, що зберігаються в БД. З′ясувати, яка саме і в якому вигляді інформація буде зберігатися в базі, на підставі цього визначити тип даних для кожного атрибуту і діапазон ймовірних значень. Розробити детальну логічну схему з дотриманням правил нормалізації.

Визначити ключові поля, з'ясувати типи відношень між пов'язаними полями. Нарисувати схему бази даних.

Отримані результати оформити як документ «Проект бази даних» засобами Word. У ньому висвітлити такі питання:

- предметна область, у якій буде використовуватися БД;
- основні задачі, які будуть розв'язуватися на основі даних, що зберігаються в БД;
- схема БД із зазначенням ключових полів, зв'язків між таблицями, їхніми видами та призначенням таблиць;
- структура кожної таблиці (імена полів, типи даних, властивості);
- зовнішні джерела даних.

Крок 2. Створення таблиць БД, заповнення таблиць економічно обґрунтованими даними

На основі виконаного на першому кроці аналізу поставленої задачі і

розробленої логічної схеми даних приступити до створення бази, починаючи зі створення таблиць за допомогою майстра. Якщо це недоцільно, то надати обґрунтування і використати конструктор. При визначенні властивостей полів необхідно звертати увагу й задавати не тільки тип та формат даних, але й додаткові властивості, такі, як значення за замовчуванням, умова на значення, маска введення (якщо це є доцільним) та ін.

Внести дані до бази в такій послідовності: спочатку заповнюються батьківські таблиці, а потім – дочірні. Створити та визначити властивості зв'язків між первинними та зовнішніми ключами таблиць.

За результатами оформити звіт, у якому:

- зазначити засоби, за допомогою яких створювалися таблиці з обґрунтуванням їхнього вибору;
- описати додаткові властивості окремих полів і таблиць у цілому (значення за замовчуванням, умови на значення, маски введення тощо);
- надрукувати дані таблиць.

Крок 3. Проектування запитів як основного інструменту розв'язання інформаційних задач

Розробити і створити запити для розв'язання задач, що сформульовані в постановці задачі: вибірки за певними умовами, визначення підсумкових даних, оновлення існуючих та видалення застарілих даних, додавання записів в існуючу або створювану таблицю. Для досягнення поставленої мети ознайомитися з різними типами запитів.

На даному кроці виконання ІНДЗ студенти повинні детально опанувати різні засоби створення запитів: за допомогою майстрів, у режимі конструктора; ознайомитися із вбудованими функціями MS Access та технологією використання будівника виразів.

Для кожної інформаційної задачі результати подати в такому вигляді:

- формулювання задачі;
- вид запиту (детальний, підсумковий тощо);
- базові джерела даних (таблиці, запити);
- перелік полів із зазначенням джерела (ім'я таблиці чи запиту), групування, сортування тощо (в разі потреби);
- обчислювані поля;
- зв'язки між джерелами даних із зазначенням полів, якщо їх встановлено під час побудови запиту;

- умови відбору (в разі потреби).

Крок 4. Проектування форм як основних елементів інтерфейсу користувача створюваного застосування (бази даних)

Виходячи із схеми даних, розробити необхідні форми для ефективного ведення бази (введення, зміна і вилучення даних таблиць).

Перебуваючи на рівні більш глибокого розуміння економічних задач, які можна розв'язувати на основі даних, що зберігаються в базі, сформулювати ці задачі додатково до тих, що були зазначені в постановці. Створити форми, за допомогою яких розв'язуватимуться вказані задачі. У них використати відповідні елементи керування, діаграми, а також зведені таблиці і зведені діаграми. Як джерела даних при цьому доцільно вживати запити, що побудовані на кроці 3.

На даному етапі виконання ІНДЗ студенти повинні ознайомитися з різними режимами роботи з формами – режимом конструктора, режимом форми або таблиці. З'ясувати види та специфічне призначення різних способів подання даних у режимі форми: в один стовпчик, стрічкова форма, зведена таблиця або зведена діаграма. Навчитися створювати складні форми: підпорядковані та пов'язані форми.

При розробці форм студенти повинні детально ознайомитися з властивостями основних елементів керування форм: полів, списків, полів із списками та кнопок.

Вміти використовувати методи візуалізації даних за допомогою побудови діаграм та настроювати властивості діаграм.

За результатами виконання кроку подати форми для виконання таких видів робіт:

- ведення БД (додавання, змінювання і вилучення даних);
- економічний аналіз, (у тому числі з використанням зведених таблиць і діаграм).
- Для кожної форми подати такі дані:
- назва форми із зазначенням виду;
- призначення;
- рисунок форми в режимі конструктора (частина копії екрану);
- рисунок форми в режимі форми (частина копії екрану).

Крок 5. Технологія створення, редагування та використання звітів у базі даних MS Access

На основі таблиць та запитів створеної бази даних розробити звіти, що відповідають документам, які використовуються у сфері діяльності відповідно до індивідуального навчально-дослідного завдання. Навчитися створювати звіти в режимі майстра і конструктора, з'ясовувати призначення різних областей звіту: заголовка, колонтитулів, заголовків груп, області даних, областей приміток звіту та груп. Навчитися формувати багаторівневі звіти, керувати рівнями групувань, створювати поля із сумою з накопиченням для організації нумерування рядків у звіті.

Ознайомитися зі стандартними виразами та навчитися створювати власні в колонтитулах звітів.

Для кожного звіту результати подати в такому вигляді:

- призначення;
- джерело даних (запит, таблиці);
- рисунок звіту в режимі конструктора (частина копії екрану);
- рисунок фрагменту звіту в режимі звіту (частина копії екрану).

Крок 6. Автоматизація та створення застосувань у БД

Виходячи з постановки задачі, розробити проект ієрархії керування у додатку (послідовність відкривання форм та звітів).

Використовуючи макроси, автоматизувати виконання операцій на формах, а саме: відбір даних із заданого діапазону, доступність елементів керування, встановлення пароля на відкривання форми і виконання інших операцій, пошук потрібного документа тощо.

Реалізувати схему керування у застосуванні з використанням кнопкових форм.

Налаштувати параметри автозапуску додатка.

За результатами оформити звіт, у який включити:

- схему ієрархії керування у застосуванні (рисунки кнопкових форм і зв'язки між ними);
- рисунки форм з елементами керування, з якими пов'язані макроси.

Крок 7. Оформлення та захист ІНДЗ

Засобами MS Word оформити загальний звіт із виконаного ІНДЗ. Для цього достатньо об'єднати й узгодити звіти, що були створені на попередніх шести кроках.

Засобами PowerPoint створити презентацію, в якій відобразити:

- постановку задачі;
- найцікавіші моменти створеного застосування;
- можливі сфери використання.

У слайдах презентації передбачити гіперпосилання для переходу з презентації в базу даних.

Захистити ІНДЗ на заняттях в аудиторії згідно з графіком викладача.

Графік виконання індивідуального завдання

Таблиця 4.3 – Графік виконання індивідуального завдання

Семестр	Тиждень	Зміст	Форма контролю
4	1 – 2	Крок 1. Постановка задачі, розробка логічної схеми БД та переліку аналітичних задач	Поточний контроль
4	3 – 4	Крок 2. Створення таблиць БД, заповнення таблиць обґрунтованими даними	Поточний контроль
4	5– 6	Крок 3. Проектування запитів як основного інструменту розв’язання інформаційних задач	Поточний контроль
4	7 – 9	Крок 4. Проектування форм як основних елементів інтерфейсу користувача створюваного застосування (бази даних)	Поточний контроль
4	10 – 12	Крок 5. Технологія створення, редагування та використання звітів у базі даних MS Access	Поточний контроль
4	13 – 15	Крок 6. Автоматизація та створення застосувань у БД	Поточний контроль
4	16 – 18	Крок 7. Оформлення та захист ІНДЗ	Поточний контроль

Захист розроблених застосувань організується у вигляді виступу студентів перед групою з демонстрацією роботи застосування в середовищі MS Access, а також за допомогою презентації в PowerPoint. Студент повинен сформулювати, для розв’язування яких задач було розроблено застосування, і продемонструвати знайдені для цього рішення; вміти відповідати на запитання інших студентів групи та викладача.

Приклад виконання та оформлення ІНДЗ

Варіант завдання «Туристична агенція»

Предметною областю для розроблюваної бази даних є надання інфо-

рмаційних та туристичних послуг населенню туристичною агенцією. Передбачається, що користувачами бази можуть бути менеджери агенції, які безпосередньо реєструють придбані населенням тури (подорожі), виконують пошук та підбір турів, а також мають змогу проводити оперативний аналіз виконаної роботи: підраховувати кількість проданих турів, зокрема кожним менеджером за певний проміжок часу, попит на певну категорію подорожей (країну).

Таким чином, основною метою розробки такої бази даних є реалізація можливості поповнення і зберігання інформації про туристичні подорожі. Виходячи з аналізу предметної області і поставлених задач, робиться висновок щодо інформації, яка буде зберігатися в базі. У даному випадку це є інформація про клієнтів агенції, наявні тури (подорожі), країни, умови проживання в готелях та транспорт, співробітників агенції, а також інформація про власне укладені договори (основна робоча таблиця бази).

За умови дотримання вимог трьох перших форм нормалізації при розподілі інформації в окремі таблиці. Логічна схема бази може мати вигляд, який показано на рис. 4.1. З цього рисунку видно, що виконаними є перший і другий кроки індивідуального завдання, а саме: визначена структура таблиць, призначені ключові поля, встановлені зв’язки між таблицями із забезпеченням підтримки цілісності даних, про що свідчить визначений тип відношення між полями таблиць.

Перелік запитів формується студентами самостійно, виходячи із конкретної теми завдання, й охоплює коло реальних задач, але запити повинні продемонструвати вміння студента побудувати всі види запитів: на вибірку з різними умовами відбору даних, із полями, що обчислюються, підсумкові запити, перехресні запити, запити з параметром, запити на зміну таблиць.

Нижче наведений перелік можливих запитів для БД «Туристична агенція». Запити на вибірку даних:

1. Знайти список прізвищ та даних по клієнтах, які придбали тури в агенції.
2. Знайти список потенційно можливих подорожей, тобто список маршрутів із зазначенням країни, курорту, кількості днів та вартості туру.
3. Знайти список укладених договорів, для яких необхідно оформлення візи (параметричний запит).
4. Обчислити вартість однієї подорожі (туру).

5. Обчислити сумарну вартість турів, проданих за одним напрямом.

6. Створити 3 підсумкових запити з використанням функцій **Sum**, **Count**, **Max**, використовуючи різні критерії для групування даних.

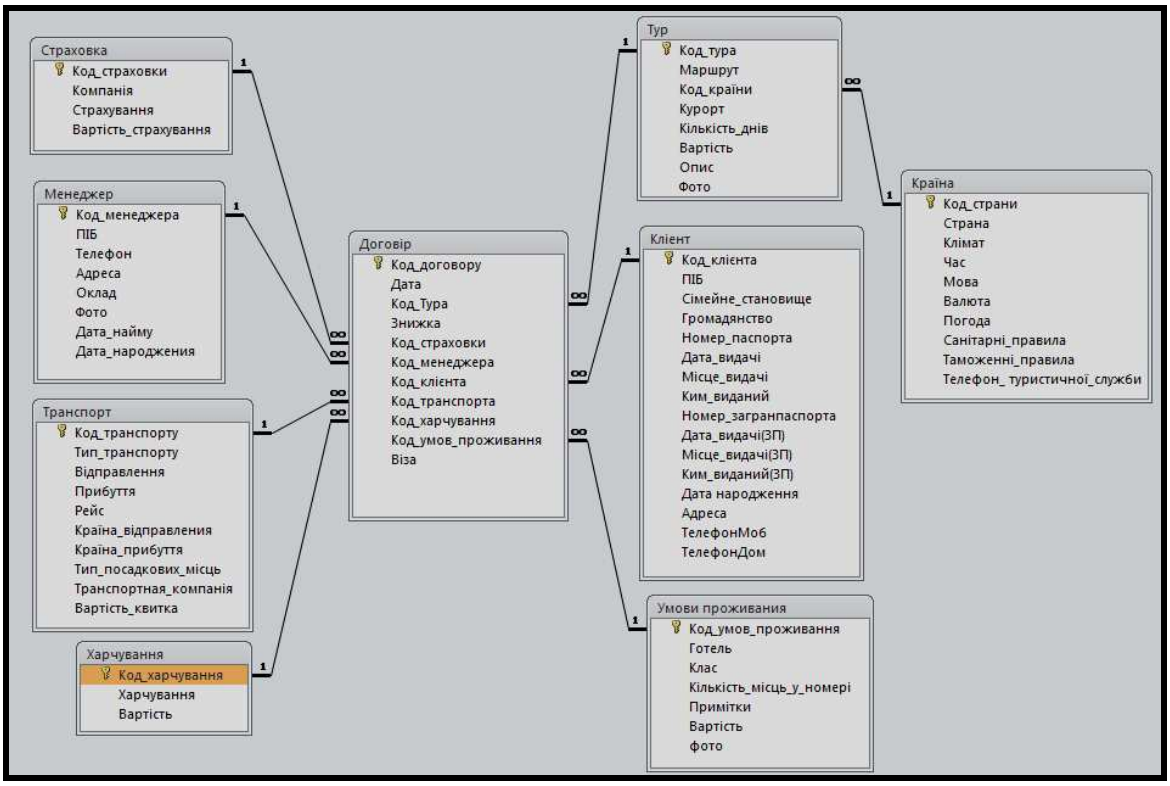


Рис. 4.1 – Завершений вигляд схеми даних для варіанта ІНДЗ «Туристична агенція»

7. Створити перехресний запит для аналізу кількості турів, проданих менеджерами в різні країни.

8. Використати модифікуючий запит на додавання, видалення та оновлення записів за заданим критерієм.

Кількість та зміст форм визначається студентами самостійно, але форми повинні задовольняти такі загальні вимоги завдання:

- Обов’язковою є головна кнопкова форма.
- Форми в різних режимах і з різними елементами керування, підпорядкована форма, зведена таблиця та діаграма, побудовані діаграми різного типу. Використання макросів з групами макрокоманд для управління й автоматизації роботи додатка.

Кількість та зміст звітів визначається студентами самостійно, але звіти повинні задовольняти загальні вимоги завдання:

- Звіт, повинен мати заголовок, примітки, групування в області да-

них із підрахунком кількості записів у групі, інші підсумки про дані групи записів.

- Питання щодо встановлення підтримки властивостей каскадного оновлення та каскадного видалення даних не є однозначним і повинно бути обґрунтованим для кожного зв'язку.
- При розробці структури таблиць студенти повинні максимально використати властивості полів різних типів з метою зменшити ризик помилок і затрати часу при введенні даних.

Наприклад, створити маски введення даних, які вимагають дотримання певного формату, наприклад номери телефонів, різноманітні шифри, поля дат та ін. Відповідні приклади показані на рис. 4.2, 4.3.

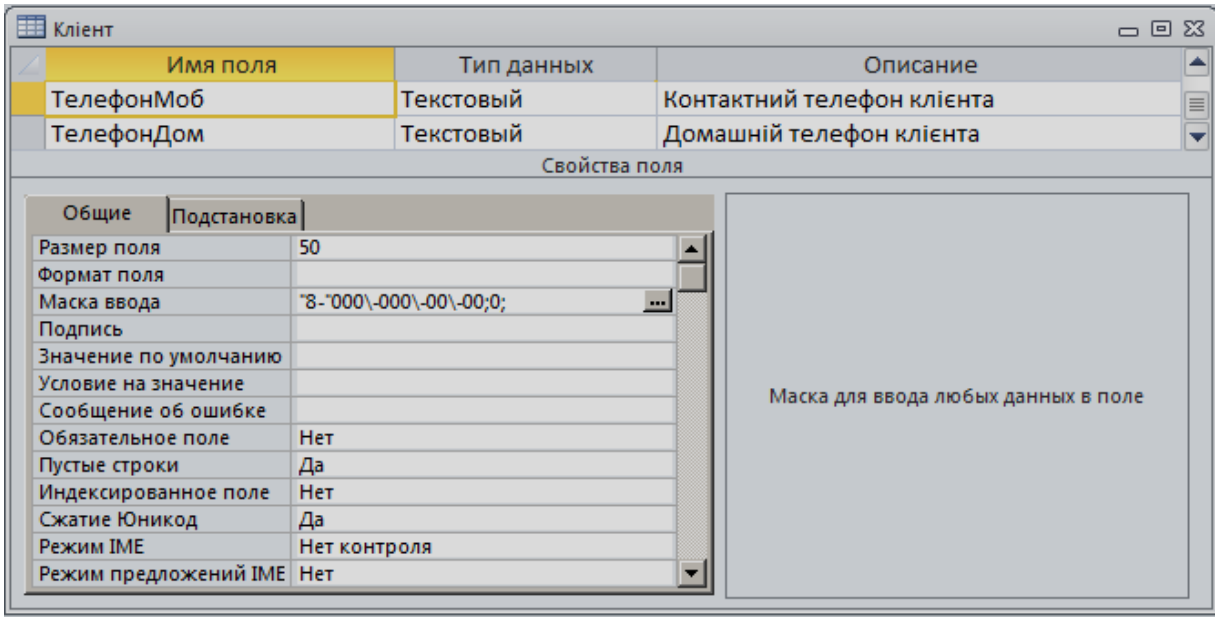


Рис. 4.2 – Використання маски введення для забезпечення єдиного формату введення номера телефону

Оскільки основним об'єктом будь-якої бази для кінцевого користувача є форма, що забезпечує його доступ до необхідної інформації, то особлива увага при виконанні ІНДЗ має бути приділена саме розробці необхідних форм із зручним інтерфейсом. Обов'язковою вимогою завдання є створення головної кнопкової форми бази, яка дає доступ до всіх інших необхідних об'єктів. Приклад головної кнопкової форми наведений на рис. 4.4. Форма створюється за допомогою диспетчера кнопкових форм і доробляється безпосередньо в конструкторі.

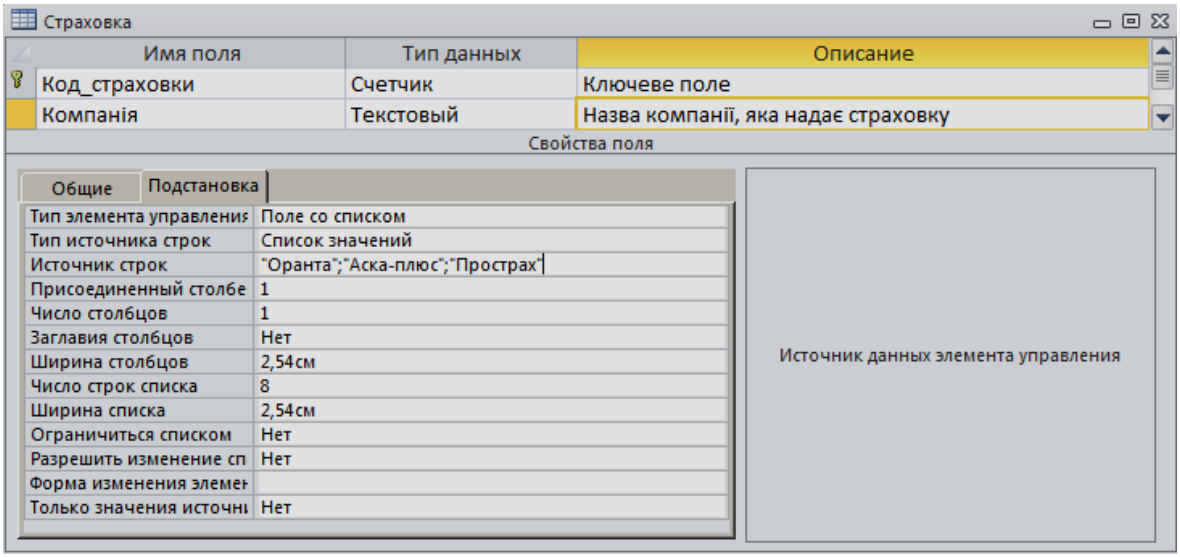


Рис. 4.3 – Створення списку підстановки для обмеження введення можливих значень поля

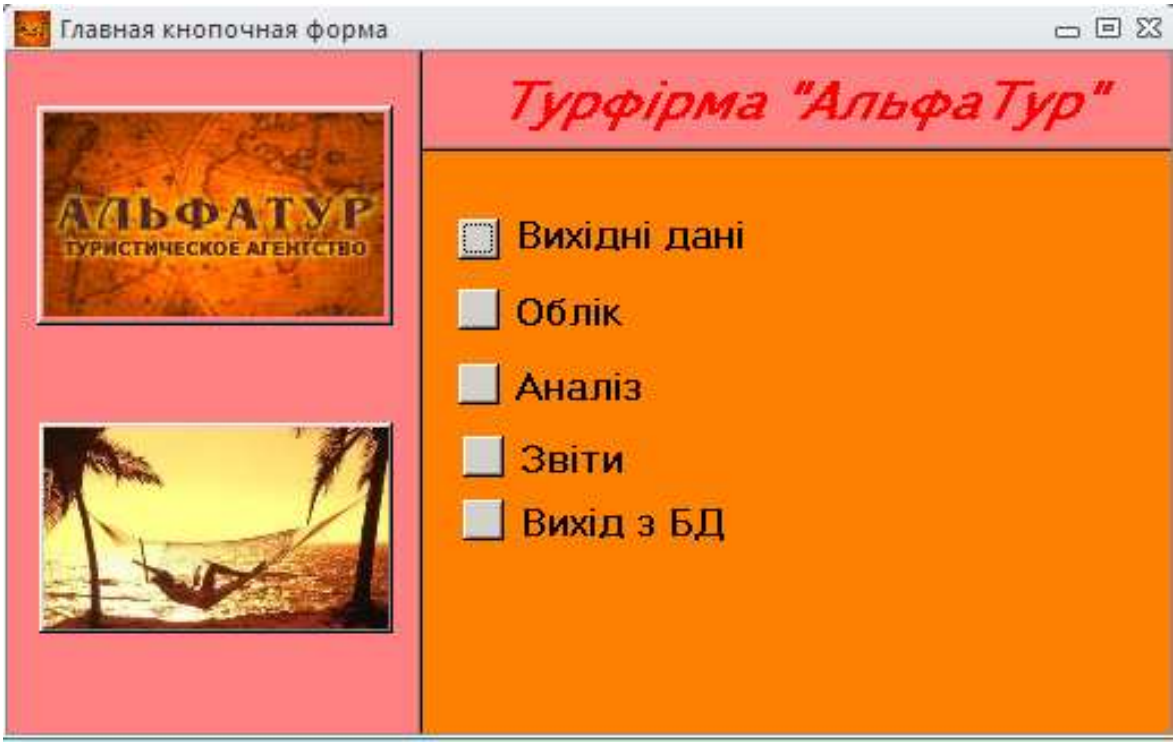


Рис. 4.4 – Приклад головної кнопкової форми для варіанта ІНДЗ «Туристична агенція»

Як видно з рис. 4.4 кнопкова форма дозволяє відкрити всі необхідні об’єкти бази і завершувати роботу з додатком. Вона також є багатосторінковою: спочатку відбувається перехід до окремої її групи об’єктів – звітів, довідників (вихідних даних), а потім до певного об’єкта.

В ІНДЗ сформульовані технічні вимоги: до режиму форми – в стовп-

чик, стрічкова, підпорядкована; до складу областей форми – область заголовку, область даних, примітки; до складу елементів управління, використаних на формі – поля, поля зі списками, списки, поля, що обчислюються, керуючі кнопки. Зміст і дизайн кожної форми студенти повинні розробити і виконати самостійно.

Зразки форм, які відповідають загальним вимогам ІНДЗ, наведені на рис. 4.5 – 4.10.

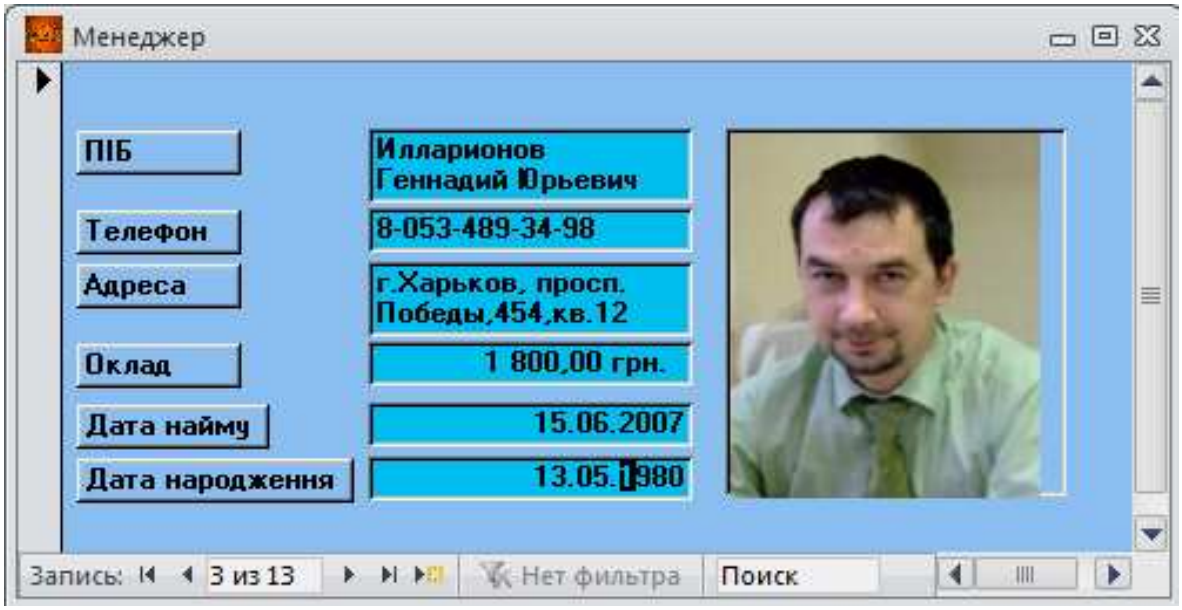


Рис. 4.5 – Зразок форми в стовпчик, яка вміщує приєднану рамку об’єкта для відображення фотографії. У таблиці цьому полю відповідає поле типу OLE

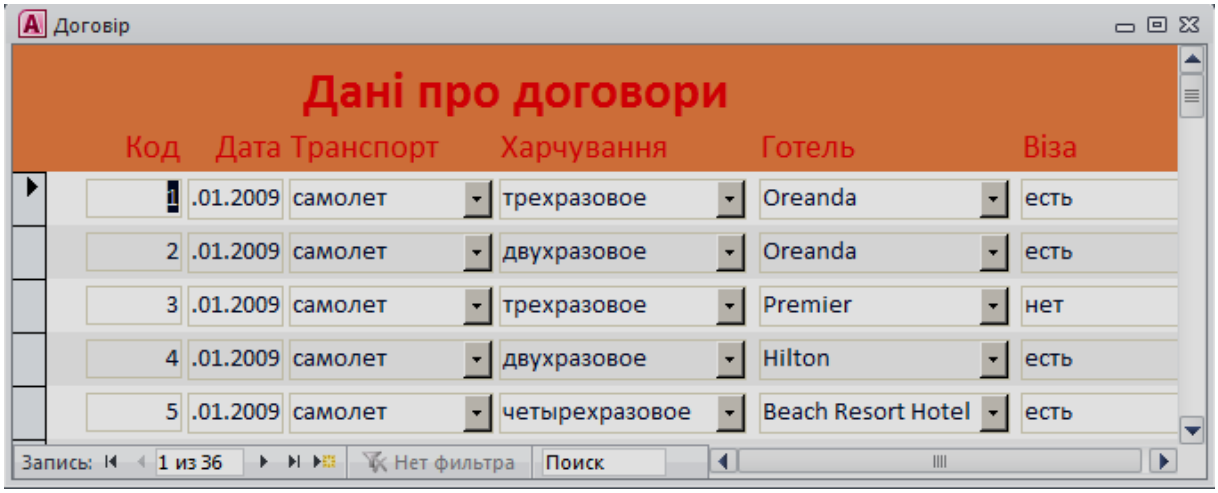


Рис. 4.6 – Зразок стрічкової форми, яка вміщує поля зі списками для забезпечення зручності введення даних

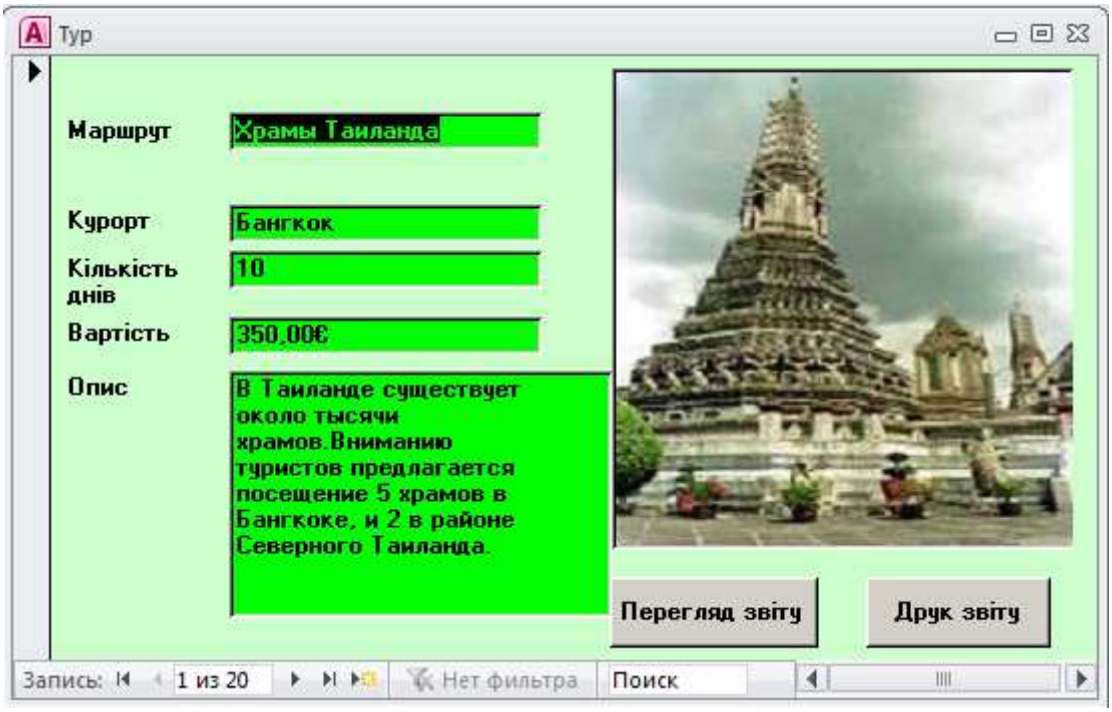


Рис. 4.7 – Зразок форми з описом туру з керуючими кнопками, які надають можливість переглянути та роздрукувати перелік турів

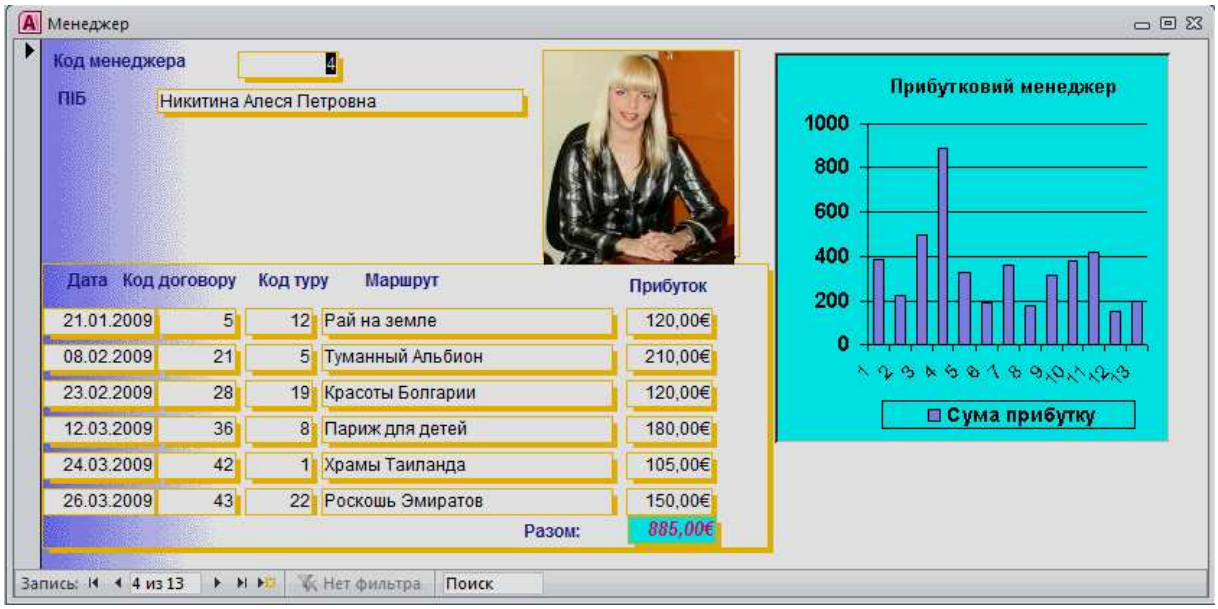


Рис. 4.8 – Зразок головної форми, що вміщує підпорядковану форму

На рис. 4.8 показано форму, яка призначена для аналізу даних. Головна форма має діаграму, яка відображає дані про прибуток, принесений кожним менеджером.

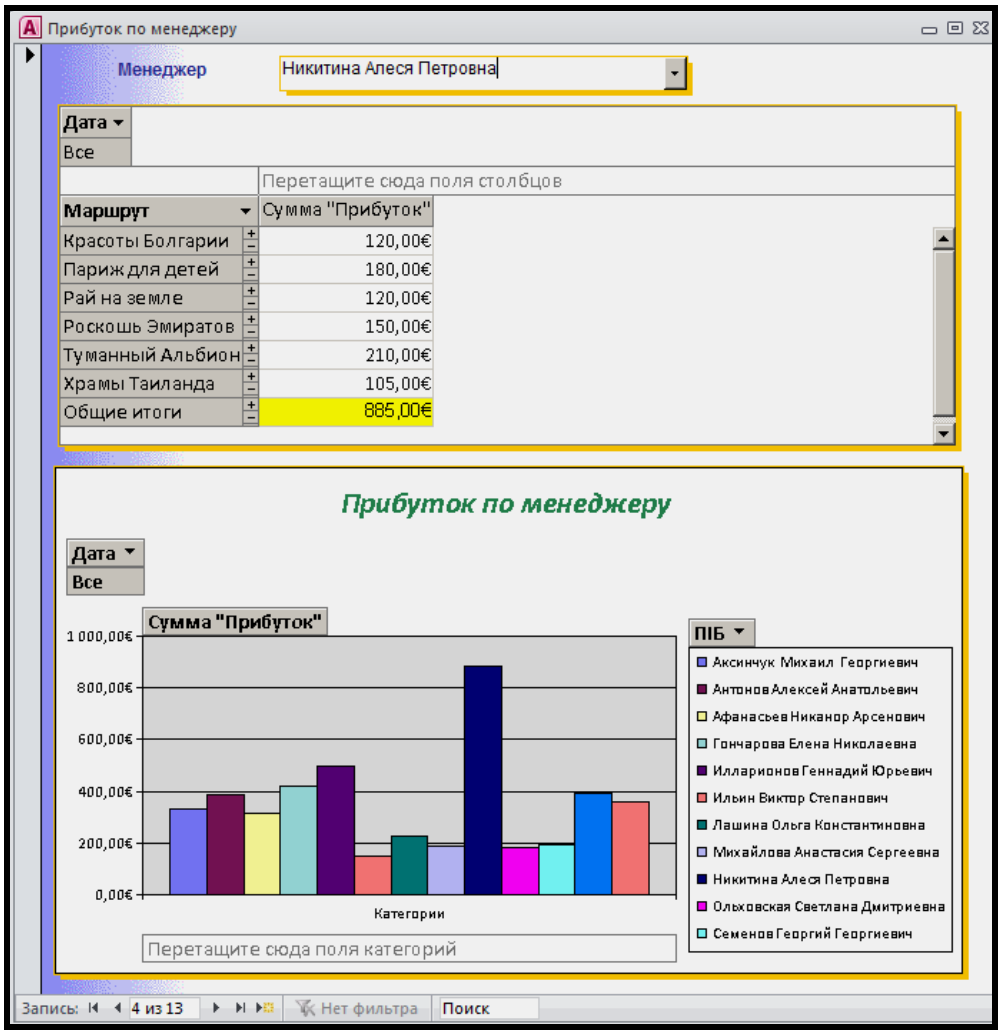


Рис. 4.9 – Зведена таблиця і зведена діаграма для підбиття підсумків за продажами окремими менеджерами

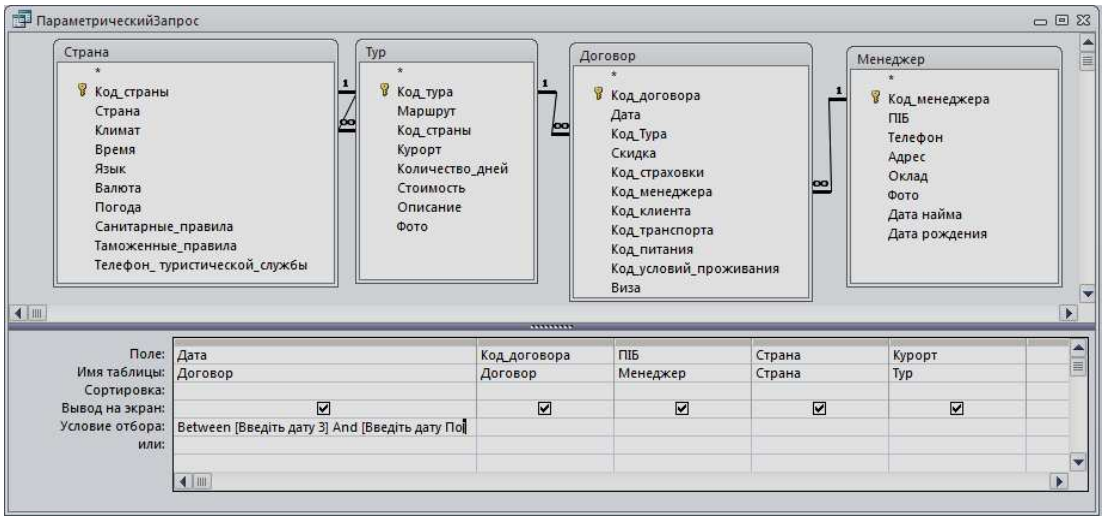


Рис. 4.10 – Використання параметра як аргументу оператору порівняння **Between** у запиті на вибірку даних

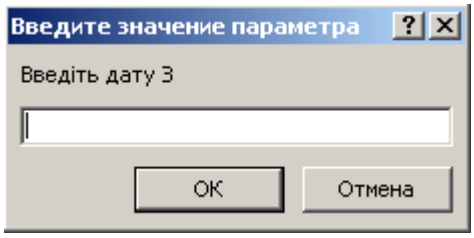


Рис. 4.11 – Введення значення параметра при виконанні параметричного запиту на вибірку даних

Работа турфірми

Місяць	Країна	Дата	ПІБ менеджера	Маршрут	Вартість
Январь 2009				*Разом за місяць:*	*62 271,50 €*
	Великобритания				
		24.01.2009	Журавская Елена Игоревна	Туманный Альбион	3 355,00€
	Египет				
		15.01.2009	Курочкин Сергей Николаевич	Лучший дайвинг	2 210,00€
		15.01.2009	Курочкин Сергей Николаевич	Египетские SPA	2 210,00€
	Индонезия				
		24.01.2009	Журавская Елена Игоревна	Индонезийский рай	3 355,00€
	Испания				
		30.01.2009	Мухина Олеся Петровна	Секреты Ибицы	2 530,00€
		31.01.2009	Мухоморова Надежда Геннадие	Секреты Ибицы	2 555,00€
	Куба				
		15.01.2009	Курочкина Александра Констант	Дух свободы	1 672,50€
		20.01.2009	Ермолаева Надежда Игоревна	Дух свободы	2 842,50€
		21.01.2009	Хирный Михаил Иванович	Рай на земле	3 320,00€

Рис. 4.12 – Зразок звіту з підбиттям підсумків за групами записів

Таким чином, основні групи задач даної предметної області були вирішені. За допомогою запитів та форм виконано аналіз даних, які зберігаються в таблицях бази даних. Створено форми, що дозволяють створювати нові записи в таблицях, модифікувати вже існуючі та виконувати навігацію по додатку. На основі таблиць та запитів БД створено звіти, що відповідають документам, які вживаються у сфері діяльності туристичної агенції.

Глосарій

Б

База даних — Сукупність взаємозалежних даних, організованих відповідно до схеми бази даних таким чином, щоб з ними міг працювати користувач

В

Вираз — Обчислює значення об'єктів бази даних. Може містити математичні й логічні операції, викликати функції, вбудовані користувачем

Вікно бази даних — Містить списки всіх об'єктів бази даних і забезпечує до них доступ

Вікно властивостей — Призначене для перегляду й зміни властивостей таблиць, запитів, форм, звітів до даних й елементів керування

Вільний елемент керування — Елемент керування у формі, звіті або на сторінці доступу до даних, не пов'язаний з даними

Властивість — Характеристика об'єкта, що може приймати певне значення

Г

Головна форма — Форма, що включає іншу форму

Групування — Поділ даних на групи за певним критерієм

Д

Дані — Довільна інформація, представлена в цифровій формі

Джерело даних — Таблиця або запит, на основі даних яких заповнюються форма, звіт, елемент керування або інший об'єкт

Джерело записів — Таблиця, запит або інструкція SQL, що забезпечує дані для форми або звіту

Діаграма — Графічний образ, що відображає кількісну залежність даних

Додаток у БД — Програма, що служить для зручної роботи з об'єктами БД

Е

Експорт даних — Копіювання об'єкта із БД

З

Записи БД	Рядки таблиці
Запити на видалення	Призначені для видалення з таблиці записів за заданою умовою
Запити на відновлення	Призначені для зміни значення полів таблиці для відібраних записів
Запити на додавання	Призначені для додавання кількох записів із запиту в таблицю
Запити на створення таблиці	Призначені для створення нової таблиці, в яку можуть додаватись записи із таблиць та запитів
Звіт	Об'єкт, призначений для виводу обраної із БД інформації у вигляді документа на принтер або монітор
Зовнішній ключ	Поле, що служить для зв'язку з іншою таблицею

І

Імена полів	Назви колонок таблиці
Імпорт даних	Копіювання об'єкта в БД

К

Каскадне видалення	Засіб підтримки цілісності даних у пов'язаних таблицях, який при видаленні запису в головній (батьківській) таблиці забезпечує видалення всіх пов'язаних записів у залежній (дочірній) таблиці
Каскадне відновлення	Засіб підтримки цілісності даних у пов'язаних таблицях, при якому в разі зміни значення ключового поля в головній (батьківській) таблиці забезпечується відновлення всіх пов'язаних записів у залежній (дочірній) таблиці
Кнопки переходу	Кнопки, які розміщені на нижній границі вікон у режимі таблиць і форм та використовують для переміщення по записах
Кнопкова форма	Форма, яка забезпечує користувачу доступ до функцій додатка

М

Макрос	Це об'єкт, призначений для автоматизації дій, які виконуються за допомогою команд меню або кнопок панелей інструментів

Модуль	Об'єкт, що становить програму, написану мовою VBA, й використовується при створенні складних додатків
	О
Обчислювальне поле	Поле в запиті або формі, у якості значення якого виводиться результат обчислення виразу, а не дані, збережені в таблиці
Операція	Дія, виконувана над даними таблиць бази даних
	П
Панель елементів	Панель інструментів, що містить кнопки, за допомогою яких створюються елементи керування у формах та звітах
Первинний ключ	Поле або сукупність полів, які однозначно визначають запис таблиці
Підлегла форма (звіт)	Підлегла форма (звіт), що вбудовується в головну форму (звіт)
Повідомлення про помилку	Властивість поля, що визначає текст повідомлення, коли порушується умова на значення
Поле зі списком	Комбінований елемент керування, що поєднує в собі текстове поле для введення й відображення даних і список для вибору значення
Поле	Дані колонок таблиці
Поле підстановок	Список можливих значень поля в таблиці, що використовується для спрощення введення даних
	Р
Реляційна БД	Дані, подані у вигляді взаємозалежних таблиць
Система керування базами даних (СКБД)	Сукупність програмних та мовних засобів, які забезпечують керування БД
Складений ключ	Первинний ключ таблиці, що складається з декількох полів
Сортування	Зміна порядку, в якому подані дані
Стрічкова форма	Вид форми, в якій виводиться одночасно кілька записів із базової таблиці або запиту у формі рядка

Т

Таблиця — Це об'єкт, що служить для визначення й зберігання даних одного виду

У

Умова на значення — Властивість поля, що визначає обмеження, які повинні задовольняти правильні значення поля

Ф

Форма — Об'єкт, що створює зручний інтерфейс із даними (однієї або декількома таблицями). Форма використовується для введення даних, відображення їх на екрані й керування роботою додатка

Ц

Цілісність даних — Система правил, що використовується для підтримки зв'язків між записами у пов'язаних таблицях

Рекомендована література

Основна

1. Гурвиц Г. А. Microsoft Access 2010. Разработка приложений на реальном примере. – Издательство: BHV, 2010. – 496с.

2. Інформатика для економістів. Навчальний посібник для студентів вищих навчальних закладів економічних спеціальностей. Беспалов В. М., Вакула А. Ю., Гострик А. М. та ін. – К. : ЦУЛ, 2003. – 788 с.

3. Мирошниченко Г. А. Реляционные базы данных: практические приемы оптимальных решений. – СПб. : БХВ-Петербург, 2005. – 400 с.

4. Єрьоміна Н. В. Проектування баз даних: Навчальний посібник. – К. : КНЕУ, 1998. – 208 с.

5. Бобцов А. А., Шиегин В. В. Банки и базы данных. Основы работы с MS Access: Ч. 1. Учебное пособие. – СПб. : 2005. – 93 с.

Додаткова

6. Джеффри Д. Ульман, Дженнифер Уидом. Введение в системы баз даннях. – Издательство «Лори», 2000. – 376 с.

7. Гринченко Н. Н. Гусев Е. В., Макаров Н. П. и др. Проектирование баз данных. СУБД Microsoft Access. – М. : Телеком, 2004. – 240 с.

8. Карпова Т. С. Базы данных: модели, разработка, реализация. – СПб. : Питер, 2001. – 304 с.

Ресурси мережі Internet

9. Сайт додатків Office корпорації Microsoft [Електронний ресурс]. – Режим доступу: http://office.microsoft.com/ru-ru/access-help/

10. Сайт комп′ютерної літератури «Таурион» [Електронний ресурс]. – Режим доступу: http://www.taurion.ru/access

11. Сайт «AccesSoft» по розробці баз даних за допомогою Access [Електронний ресурс]. – Режим доступу: http://www.accessoft.ru/index.html

12. Сайт «Библиотека on-line» по розробці баз даних [Електронний ресурс]. – Режим доступу: http://citforum.ru/database

Навчальне видання

ТОЛСТОХАТЬКО Віктор Антонович,
ПОМОРЦЕВА Олена Євгенівна,
ПАТРАКЕЄВ Ігор Михайлович

БАЗИ ДАНИХ: ПРОЕКТУВАННЯ ТА ВИКОРИСТАННЯ ДЛЯ ОБЛІКУ НЕРУХОМОГО МАЙНА

НАВЧАЛЬНИЙ ПОСІБНИК

(для студентів денної та заочної форми навчання
напряму підготовки 6.080101 «Геодезія, картографія та землеустрій»)

За авторською редакцією
Відповідальний за випуск *К. А. Мамонов*
Комп'ютерне верстання *Є. Г. Панова*
Дизайн обкладинки *І. П. Шелехов*

Підп. до друку 12.06.2014 р. Формат 60x84/16
Друк на ризографі Ум. друк. арк. 7
Тираж 500 пр. Зам. №

Видавець і виготовлювач:
Харківський національний університет міського господарства імені О. М. Бекетова,
вул. Революції,12, Харків, 61002
Електронна адреса: rectorat@kname.edu.ua
Свідоцтво суб'єкта видавничої справи:
ДК № 4705 від 28.03.2014